



(12) 发明专利

(10) 授权公告号 CN 111597015 B

(45) 授权公告日 2023. 01. 06

(21) 申请号 202010343305.8

(22) 申请日 2020.04.27

(65) 同一申请的已公布的文献号
申请公布号 CN 111597015 A

(43) 申请公布日 2020.08.28

(73) 专利权人 腾讯科技(深圳)有限公司
地址 518057 广东省深圳市南山区高新区
科技中一路腾讯大厦35层
专利权人 中国人民大学

(72) 发明人 李海翔 卢卫 杜小勇 张孝
赵展浩 赵泓尧 潘安群

(74) 专利代理机构 北京三高永信知识产权代理
有限责任公司 11138
专利代理师 祝亚男

(51) Int.Cl.

G06F 9/46 (2006.01)

G06F 16/27 (2019.01)

(56) 对比文件

CN 109977171 A, 2019.07.05

US 2011/0302143 A1, 2011.12.08

审查员 崔景妹

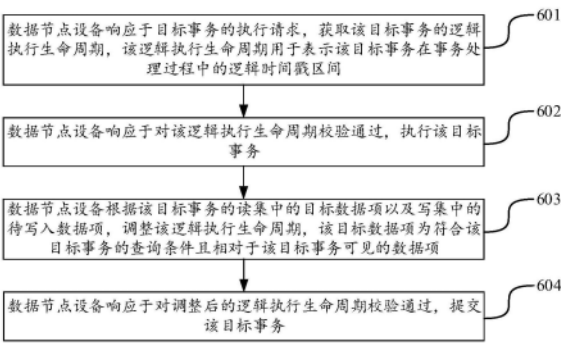
权利要求书5页 说明书36页 附图5页

(54) 发明名称

事务处理方法、装置、计算机设备及存储介
质

(57) 摘要

本申请公开了一种事务处理方法、装置、计算机设备及存储介质,属于数据库技术领域。本申请通过响应于目标事务的执行请求,获取目标事务的逻辑执行生命周期,在对该逻辑执行生命周期校验通过之后,执行目标事务,在事务执行或者验证阶段中,根据目标事务的读集中的目标数据项以及写集中的待写入数据项,对逻辑执行生命周期进行调整,在对调整后的逻辑执行生命周期校验通过之后,提交目标事务,这种基于逻辑执行生命周期来处理事务的机制,无需依赖于锁机制,因此更加适用于读多写少的互联网应用场景,有利于提升分布式数据库系统的事务吞吐量,有利于对系统进行扩容,能够提升系统的事务处理性能。



1. 一种事务处理方法,其特征在于,所述方法包括:

响应于目标事务的执行请求,获取所述目标事务的逻辑执行生命周期,所述逻辑执行生命周期用于表示所述目标事务在事务处理过程中的逻辑时间戳区间;

响应于对所述逻辑执行生命周期校验通过,执行所述目标事务;

根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期,所述目标数据项为符合所述目标事务的查询条件且相对于所述目标事务可见的数据项,其中,对所述写集中任一待写入数据项的读事务列表中的任一读事务,响应于所述读事务处于正在运行状态,在不同的可串行化隔离级别下,对所述逻辑执行生命周期进行不同的调整,所述在不同的可串行化隔离级别下,对所述逻辑执行生命周期进行不同的调整包括:响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,在等待所述读事务结束后继续处理所述目标事务,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳上界;响应于数据库系统处于线性点可串行化隔离级别,回滚所述目标事务;响应于数据库系统处于顺序并发结果可串行化隔离级别,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳下界,将所述读事务的时间戳上界调整至小于或等于所述逻辑执行生命周期的时间戳下界;

响应于对调整后的逻辑执行生命周期校验通过,提交所述目标事务;

其中,所述数据库系统处于线性可串行化隔离级别是指并发事务和并行事务均保持全序;所述数据库系统处于线性并发结果可串行化隔离级别是指并发事务按可串行化顺序,保持全序,并行事务按提交顺序,在全局系统范围内保持全序;所述数据库系统处于顺序可串行化隔离级别是指并发事务之间保证可串行化,且所有事务之间存在全序关系;所述数据库系统处于线性点可串行化隔离级别是指并发事务按可串行化顺序,保持全序,并行事务按提交顺序,在全局系统范围内保持全序,每个事务的开始时间和结束时间在同一个真实时间点上;所述数据库系统处于顺序并发结果可串行化隔离级别是指并发事务执行结果与确定的顺序执行序列等价,并发事务的事务序确定了并发事务的全局序,而并行事务不存在全局序。

2. 根据权利要求1所述的方法,其特征在于,若所述目标事务涉及针对数据项的读取操作,所述执行所述目标事务包括:

基于所述执行请求中的查询条件,确定所述查询条件所对应的至少一个待判断数据项;

从所述至少一个待判断数据项中,确定相对于所述目标事务可见的目标数据项,将所述目标数据项存储到所述目标事务的读集中。

3. 根据权利要求2所述的方法,其特征在于,所述从所述至少一个待判断数据项中,确定相对于所述目标事务可见的目标数据项包括:

响应于数据库系统处于顺序并发结果可串行化隔离级别,对所述至少一个待判断数据项中任一待判断数据项,若产生所述待判断数据项的事务的逻辑提交时间戳小于所述目标事务的事务快照时间戳,确定所述待判断数据项为候选数据项;将具有相同主键标识的候选数据项中逻辑提交时间戳最大的候选数据项确定为目标数据项;

其中,在所述顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周

期用于确定所述两个并发事务的先后关系,所述逻辑提交时间戳用于表示事务的逻辑提交时刻,所述事务快照时间戳用于表示事务的全局开始时刻。

4. 根据权利要求2所述的方法,其特征在于,所述从所述至少一个待判断数据项中,确定相对于所述目标事务可见的目标数据项包括:

响应于数据库系统处于除了顺序并发结果可串行化隔离级别之外的可串行化隔离级别,对所述至少一个待判断数据项中任一待判断数据项,若产生所述待判断数据项的事务的全局提交时间戳小于所述目标事务的事务快照时间戳,确定所述待判断数据项为候选数据项;将具有相同主键标识的候选数据项中全局提交时间戳最大的候选数据项确定为目标数据项;

其中,在所述顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定所述两个并发事务的先后关系,所述全局提交时间戳用于表示事务的全局提交时刻,所述事务快照时间戳用于表示事务的全局开始时刻。

5. 根据权利要求1所述的方法,其特征在于,若所述目标事务涉及针对数据项的写入操作,所述执行所述目标事务包括:

根据所述执行请求,生成待写入数据项,将所述待写入数据项存储到所述目标事务的写集中。

6. 根据权利要求1至5中任一项所述的方法,其特征在于,所述响应于对调整后的逻辑执行生命周期的冲突验证通过,提交所述目标事务之前,所述方法还包括:

响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,若所述目标事务的写集中包括与所述目标数据项具有相同主键标识的待写入数据项,回滚所述目标事务。

7. 根据权利要求1所述的方法,其特征在于,所述根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期包括:

将所述逻辑执行生命周期的时间戳下界调整至大于产生所述目标数据项的事务的逻辑提交时间戳,所述逻辑提交时间戳用于表示事务的逻辑提交时刻;

将所述逻辑执行生命周期的时间戳上界调整至小于或等于产生第一数据项的事务的逻辑提交时间戳,所述第一数据项为与所述目标数据项具有相同主键标识的下一数据项。

8. 根据权利要求1所述的方法,其特征在于,所述根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期包括:

响应于所述目标数据项对应的待写事务不为空,将所述逻辑执行生命周期的时间戳上界调整至小于或等于所述待写事务的时间戳下界。

9. 根据权利要求1所述的方法,其特征在于,所述根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期包括:

将所述逻辑执行生命周期的时间戳下界调整至大于或等于各个待写入数据项的最大读事务时间戳中的最大值,所述最大读事务时间戳用于表示读取过所述待写入数据项的各事务的逻辑提交时间戳中的最大值。

10. 根据权利要求1所述的方法,其特征在于,所述根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期包括:

对所述写集中任一待写入数据项的读事务列表中的任一读事务,响应于所述读事务处

于验证通过状态或提交完成状态,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳上界。

11. 根据权利要求1所述的方法,其特征在于,所述逻辑执行生命周期的校验过程包括:

响应于所述逻辑执行生命周期的时间戳下界小于或等于所述逻辑执行生命周期的时间戳上界,确定对所述逻辑执行生命周期校验通过;

响应于所述逻辑执行生命周期的时间戳下界大于所述逻辑执行生命周期的时间戳上界,确定对所述逻辑执行生命周期校验不通过。

12. 一种事务处理装置,其特征在于,所述装置包括:

获取模块,用于响应于目标事务的执行请求,获取所述目标事务的逻辑执行生命周期,所述逻辑执行生命周期用于表示所述目标事务在事务处理过程中的逻辑时间戳区间;

执行模块,用于响应于对所述逻辑执行生命周期校验通过,执行所述目标事务;

调整模块,用于根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期,所述目标数据项为符合所述目标事务的查询条件且相对于所述目标事务可见的数据项,其中,所述调整模块包括第二调整单元,用于对所述写集中任一待写入数据项的读事务列表中的任一读事务,响应于所述读事务处于正在运行状态,在不同的可串行化隔离级别下,对所述逻辑执行生命周期进行不同的调整,所述在不同的可串行化隔离级别下,对所述逻辑执行生命周期进行不同的调整包括:响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,在等待所述读事务结束后继续处理所述目标事务,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳上界;响应于数据库系统处于线性点可串行化隔离级别,回滚所述目标事务;响应于数据库系统处于顺序并发结果可串行化隔离级别,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳下界,将所述读事务的时间戳上界调整至小于或等于所述逻辑执行生命周期的时间戳下界;

提交模块,用于响应于对调整后的逻辑执行生命周期校验通过,提交所述目标事务;

其中,所述数据库系统处于线性可串行化隔离级别是指并发事务和并行事务均保持全序;所述数据库系统处于线性并发结果可串行化隔离级别是指并发事务按可串行化顺序,保持全序,并行事务按提交顺序,在全局系统范围内保持全序;所述数据库系统处于顺序可串行化隔离级别是指并发事务之间保证可串行化,且所有事务之间存在全序关系;所述数据库系统处于线性点可串行化隔离级别是指并发事务按可串行化顺序,保持全序,并行事务按提交顺序,在全局系统范围内保持全序,每个事务的开始时间和结束时间在同一个真实时间点上;所述数据库系统处于顺序并发结果可串行化隔离级别是指并发事务执行结果与确定的顺序执行序列等价,并发事务的事务序确定了并发事务的全局序,而并行事务不存在全局序。

13. 根据权利要求12所述的装置,其特征在于,若所述目标事务涉及针对数据项的读取操作,所述执行模块包括:

确定单元,用于基于所述执行请求中的查询条件,确定所述查询条件所对应的至少一个待判断数据项;

确定存储单元,用于从所述至少一个待判断数据项中,确定相对于所述目标事务可见的目标数据项,将所述目标数据项存储到所述目标事务的读集中。

14. 根据权利要求13所述的装置,其特征在于,所述确定存储单元用于:

响应于数据库系统处于顺序并发结果可串行化隔离级别,对所述至少一个待判断数据项中任一待判断数据项,若产生所述待判断数据项的事务的逻辑提交时间戳小于所述目标事务的事务快照时间戳,确定所述待判断数据项为候选数据项;将具有相同主键标识的候选数据项中逻辑提交时间戳最大的候选数据项确定为目标数据项;

其中,在所述顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定所述两个并发事务的先后关系,所述逻辑提交时间戳用于表示事务的逻辑提交时刻,所述事务快照时间戳用于表示事务的全局开始时刻。

15. 根据权利要求13所述的装置,其特征在于,所述确定存储单元用于:

响应于数据库系统处于除了顺序并发结果可串行化隔离级别之外的可串行化隔离级别,对所述至少一个待判断数据项中任一待判断数据项,若产生所述待判断数据项的事务的全局提交时间戳小于所述目标事务的事务快照时间戳,确定所述待判断数据项为候选数据项;将具有相同主键标识的候选数据项中全局提交时间戳最大的候选数据项确定为目标数据项;

其中,在所述顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定所述两个并发事务的先后关系,所述全局提交时间戳用于表示事务的全局提交时刻,所述事务快照时间戳用于表示事务的全局开始时刻。

16. 根据权利要求12所述的装置,其特征在于,若所述目标事务涉及针对数据项的写入操作,所述执行模块用于:

根据所述执行请求,生成待写入数据项,将所述待写入数据项存储到所述目标事务的写集中。

17. 根据权利要求12至16中任一项所述的装置,其特征在于,所述装置还包括:

回滚模块,用于响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,若所述目标事务的写集中包括与所述目标数据项具有相同主键标识的待写入数据项,回滚所述目标事务。

18. 根据权利要求12所述的装置,其特征在于,所述调整模块用于:

将所述逻辑执行生命周期的时间戳下界调整至大于产生所述目标数据项的事务的逻辑提交时间戳,所述逻辑提交时间戳用于表示事务的逻辑提交时刻;

将所述逻辑执行生命周期的时间戳上界调整至小于或等于产生第一数据项的事务的逻辑提交时间戳,所述第一数据项为与所述目标数据项具有相同主键标识的下一数据项。

19. 根据权利要求12所述的装置,其特征在于,所述调整模块用于:

响应于所述目标数据项对应的待写事务不为空,将所述逻辑执行生命周期的时间戳上界调整至小于或等于所述待写事务的时间戳下界。

20. 根据权利要求12所述的装置,其特征在于,所述调整模块用于:

将所述逻辑执行生命周期的时间戳下界调整至大于或等于各个待写入数据项的最大读事务时间戳中的最大值,所述最大读事务时间戳用于表示读取过所述待写入数据项的各事务的逻辑提交时间戳中的最大值。

21. 根据权利要求12所述的装置,其特征在于,所述调整模块包括:

第一调整单元,用于对所述写集中任一待写入数据项的读事务列表中的任一读事务,

响应于所述读事务处于验证通过状态或提交完成状态,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳上界。

22.根据权利要求12所述的装置,其特征在于,所述逻辑执行生命周期的校验过程包括:

响应于所述逻辑执行生命周期的时间戳下界小于或等于所述逻辑执行生命周期的时间戳上界,确定对所述逻辑执行生命周期校验通过;

响应于所述逻辑执行生命周期的时间戳下界大于所述逻辑执行生命周期的时间戳上界,确定对所述逻辑执行生命周期校验不通过。

23.一种计算机设备,其特征在于,所述计算机设备包括一个或多个处理器和一个或多个存储器,所述一个或多个存储器中存储有至少一条程序代码,所述至少一条程序代码由所述一个或多个处理器加载并执行以实现如权利要求1至权利要求11任一项所述的事务处理方法所执行的操作。

24.一种存储介质,其特征在于,所述存储介质中存储有至少一条程序代码,所述至少一条程序代码由处理器加载并执行以实现如权利要求1至权利要求11任一项所述的事务处理方法所执行的操作。

事务处理方法、装置、计算机设备及存储介质

技术领域

[0001] 本申请涉及数据库技术领域,特别涉及一种事务处理方法、装置、计算机设备及存储介质。

背景技术

[0002] 随着数据库技术的发展,为了能够适应大数据、云计算等业务场景,分布式数据库系统逐渐变得普及。在分布式数据库系统中进行分布式事务处理时,可以采取去中心化的事务处理技术。去中心化的事务处理技术是指,在数据库系统中不存在某一节点设备集中对事务进行协调,而是在数据库系统中存在多个节点设备能够用于充当事务协调者的角色,由于涉及到多个协调节点设备共同处理分布式事务,在对事务进行并发控制时,普遍依赖于锁机制和时间戳排序机制,而由于目前主流的互联网应用场景中存在“读请求较多、写请求较少”(简称为读多写少)的现象,上述两种机制在读多写少的场景下限制了整个分布式数据库系统的事务吞吐量的提升,因此具有较差的事务处理性能。

发明内容

[0003] 本申请实施例提供了一种事务处理方法、装置、计算机设备及存储介质,能够提升分布式数据库系统的事务吞吐量,提升分布式数据库系统的事务处理性能。该技术方案如下:

[0004] 一方面,提供了一种事务处理方法,该方法包括:

[0005] 响应于目标事务的执行请求,获取所述目标事务的逻辑执行生命周期,所述逻辑执行生命周期用于表示所述目标事务在事务处理过程中的逻辑时间戳区间;

[0006] 响应于对所述逻辑执行生命周期校验通过,执行所述目标事务;

[0007] 根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期,所述目标数据项为符合所述目标事务的查询条件且相对于所述目标事务可见的数据项;

[0008] 响应于对调整后的逻辑执行生命周期校验通过,提交所述目标事务。

[0009] 一方面,提供了一种事务处理装置,该装置包括:

[0010] 获取模块,用于响应于目标事务的执行请求,获取所述目标事务的逻辑执行生命周期,所述逻辑执行生命周期用于表示所述目标事务在事务处理过程中的逻辑时间戳区间;

[0011] 执行模块,用于响应于对所述逻辑执行生命周期校验通过,执行所述目标事务;

[0012] 调整模块,用于根据所述目标事务的读集中的目标数据项以及写集中的待写入数据项,调整所述逻辑执行生命周期,所述目标数据项为符合所述目标事务的查询条件且相对于所述目标事务可见的数据项;

[0013] 提交模块,用于响应于对调整后的逻辑执行生命周期校验通过,提交所述目标事务。

[0014] 在一种可能实施方式中,若所述目标事务涉及针对数据项的读取操作,所述执行模块包括:

[0015] 确定单元,用于基于所述执行请求中的查询条件,确定所述查询条件所对应的至少一个待判断数据项;

[0016] 确定存储单元,用于从所述至少一个待判断数据项中,确定相对于所述目标事务可见的目标数据项,将所述目标数据项存储到所述目标事务的读集中。

[0017] 在一种可能实施方式中,所述确定存储单元用于:

[0018] 响应于数据库系统处于顺序并发结果可串行化隔离级别,对所述至少一个待判断数据项中任一待判断数据项,若产生所述待判断数据项的事务的逻辑提交时间戳小于所述目标事务的事务快照时间戳,确定所述待判断数据项为候选数据项;将具有相同主键标识的候选数据项中逻辑提交时间戳最大的候选数据项确定为目标数据项;

[0019] 其中,在所述顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定所述两个并发事务的先后关系,所述逻辑提交时间戳用于表示事务的逻辑提交时刻,所述事务快照时间戳用于表示事务的全局开始时刻。

[0020] 在一种可能实施方式中,所述确定存储单元用于:

[0021] 响应于数据库系统处于除了顺序并发结果可串行化隔离级别之外的可串行化隔离级别,对所述至少一个待判断数据项中任一待判断数据项,若产生所述待判断数据项的事务的全局提交时间戳小于所述目标事务的事务快照时间戳,确定所述待判断数据项为候选数据项;将具有相同主键标识的候选数据项中全局提交时间戳最大的候选数据项确定为目标数据项;

[0022] 其中,在所述顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定所述两个并发事务的先后关系,所述全局提交时间戳用于表示事务的全局提交时刻,所述事务快照时间戳用于表示事务的全局开始时刻。

[0023] 在一种可能实施方式中,若所述目标事务涉及针对数据项的写入操作,所述执行模块用于:

[0024] 根据所述执行请求,生成待写入数据项,将所述待写入数据项存储到所述目标事务的写集中。

[0025] 在一种可能实施方式中,所述装置还包括:

[0026] 回滚模块,用于响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,若所述目标事务的写集中包括与所述目标数据项具有相同主键标识的待写入数据项,回滚所述目标事务。

[0027] 在一种可能实施方式中,所述调整模块用于:

[0028] 将所述逻辑执行生命周期的时间戳下界调整至大于产生所述目标数据项的事务的逻辑提交时间戳,所述逻辑提交时间戳用于表示事务的逻辑提交时刻;

[0029] 将所述逻辑执行生命周期的时间戳上界调整至小于或等于产生第一数据项的事务的逻辑提交时间戳,所述第一数据项为与所述目标数据项具有相同主键标识的下一数据项。

[0030] 在一种可能实施方式中,所述调整模块用于:

[0031] 响应于所述目标数据项对应的待写事务不为空,将所述逻辑执行生命周期的时间

戳上界调整至小于或等于所述待写事务的时间戳下界。

[0032] 在一种可能实施方式中,所述调整模块用于:

[0033] 将所述逻辑执行生命周期的时间戳下界调整至大于或等于各个待写入数据项的最大读事务时间戳中的最大值,所述最大读事务时间戳用于表示读取过所述待写入数据项的各事务的逻辑提交时间戳中的最大值。

[0034] 在一种可能实施方式中,所述调整模块包括:

[0035] 第一调整单元,用于对所述写集中任一待写入数据项的读事务列表中的任一读事务,响应于所述读事务处于验证通过状态或提交完成状态,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳上界;

[0036] 第二调整单元,用于响应于所述读事务处于正在运行状态,在不同的可串行化隔离级别下,对所述逻辑执行生命周期进行不同的调整。

[0037] 在一种可能实施方式中,所述第二调整单元用于:

[0038] 响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,在等待所述读事务结束后继续处理所述目标事务,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳上界;

[0039] 响应于数据库系统处于线性点可串行化隔离级别,回滚所述目标事务;

[0040] 响应于数据库系统处于顺序并发结果可串行化隔离级别,将所述逻辑执行生命周期的时间戳下界调整至大于或等于所述读事务的时间戳下界,将所述读事务的时间戳上界调整至小于或等于所述逻辑执行生命周期的时间戳下界。

[0041] 在一种可能实施方式中,所述逻辑执行生命周期的校验过程包括:

[0042] 响应于所述逻辑执行生命周期的时间戳下界小于或等于所述逻辑执行生命周期的时间戳上界,确定对所述逻辑执行生命周期校验通过;

[0043] 响应于所述逻辑执行生命周期的时间戳下界大于所述逻辑执行生命周期的时间戳上界,确定对所述逻辑执行生命周期校验不通过。

[0044] 一方面,提供了一种计算机设备,该计算机设备包括一个或多个处理器和一个或多个存储器,该一个或多个存储器中存储有至少一条程序代码,该至少一条程序代码由该一个或多个处理器加载并执行以实现如上述任一种可能实现方式的事务处理方法所执行的操作。

[0045] 一方面,提供了一种存储介质,该存储介质中存储有至少一条程序代码,该至少一条程序代码由处理器加载并执行以实现如上述任一种可能实现方式的事务处理方法所执行的操作。

[0046] 本申请实施例提供的技术方案带来的有益效果至少包括:

[0047] 通过响应于目标事务的执行请求,获取目标事务的逻辑执行生命周期,在对该逻辑执行生命周期校验通过之后,执行目标事务,在事务执行或者验证阶段中,根据目标事务的读集中的目标数据项以及写集中的待写入数据项,对逻辑执行生命周期进行调整,在对调整后的逻辑执行生命周期校验通过之后,提交目标事务,这种基于逻辑执行生命周期来处理事务的机制,无需依赖于锁机制,因此更加适用于读多写少的互联网应用场景,有利于提升分布式数据库系统的事务吞吐量,能够提升整个分布式数据库系统的事务处理性能。

附图说明

[0048] 为了更清楚地说明本申请实施例中的技术方案,下面将对实施例描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本申请的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0049] 图1是本申请实施例提供的一种事务处理方法的实施环境示意图;

[0050] 图2是本申请实施例提供的一种事务执行顺序的原理性示意图;

[0051] 图3是本申请实施例提供的一种事务执行顺序的原理性示意图;

[0052] 图4是本申请实施例提供的一种数据项结构的原理性示意图;

[0053] 图5是本申请实施例提供的一种事务处理方法的交互流程图;

[0054] 图6是本申请实施例提供的一种事务处理方法的流程图;

[0055] 图7是本申请实施例提供的一种事务处理方法的交互流程图;

[0056] 图8是本申请实施例提供的一种事务处理装置的结构示意图;

[0057] 图9是本申请实施例提供的一种计算机设备的结构示意图。

具体实施方式

[0058] 为使本申请的目的、技术方案和优点更加清楚,下面将结合附图对本申请实施方式作进一步地详细描述。

[0059] 本申请中术语“第一”“第二”等字样用于对作用和功能基本相同的相同项或相似项进行区分,应理解,“第一”、“第二”、“第n”之间不具有逻辑或时序上的依赖关系,也不对数量和执行顺序进行限定。

[0060] 本申请中术语“至少一个”是指一个或多个,“多个”的含义是指两个或两个以上,例如,多个第一位置是指两个或两个以上的第一位置。

[0061] 在介绍本申请实施例之前,需要引入一些云技术领域内的基本概念:

[0062] 云技术(Cloud Technology):是指在广域网或局域网内将硬件、软件、网络等系列资源统一起来,实现数据的计算、储存、处理和共享的一种托管技术,也即是基于云计算商业模式应用的网络技术、信息技术、整合技术、管理平台技术、应用技术等的总称,可以组成资源池,按需所用,灵活便利。云计算技术将变成云技术领域的重要支撑。技术网络系统的后台服务需要大量的计算、存储资源,如视频网站、图片类网站和更多的门户网站。伴随着互联网行业的高度发展和应用,将来每个物品都有可能存在自己的识别标志,都需要传输到后台系统进行逻辑处理,不同程度级别的数据将会分开处理,各类行业数据皆需要强大的系统后盾支撑,均能通过云计算来实现。

[0063] 云存储(Cloud Storage):是在云计算概念上延伸和发展出来的一个新的概念,分布式云存储系统(以下简称存储系统)是指通过集群应用、网格技术以及分布存储文件系统等功能,将网络中大量各种不同类型的存储设备(存储设备也称之为存储节点)通过应用软件或应用接口集合起来协同工作,共同对外提供数据存储和业务访问功能的一个存储系统。

[0064] 数据库(Database):简而言之可视为一种电子化的文件柜——存储电子文件的处所,用户可以对文件中的数据进行新增、查询、更新、删除等操作。所谓“数据库”是以一定方

式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。

[0065] 数据的全态(Full State):对于数据库系统中的数据项,基于状态属性的不同,可以划分为三种状态:当前态、过渡态和历史态,该三种状态合称为“数据的全态”,简称全态数据,全态数据中的各个不同状态属性,可以用于标识数据在其生命周期轨迹中所处的状态。

[0066] 1、当前态(Current State):最新版本的数据项,是处于当前阶段的数据项。

[0067] 2、历史态(Historical State):数据项在历史上的一个状态,其值是旧值,不是当前值。多个历史态数据项可以对应于同一主键标识,反映了具有该主键标识的各个数据项的状态变迁的过程。处于历史态的数据项,只能被读取而不能被修改或删除。

[0068] 3、过渡态(Transitional State):不是当前态数据项也不是历史态数据项,处于从当前态向历史态转变的过程中,这种处于过渡态的数据也称为半衰数据。

[0069] 基于上述名词解释,不同的数据项可以具有相同的主键标识(Primary Key,PK),此时,具有相同主键标识的各个数据项可以构成一个全态数据集,该全态数据集内的各个数据项在本质上用于表示全态数据,也即是说,在对具有该主键标识的初始数据项进行多次修改(或删除)的过程中,由于修改(或删除)时刻不同而产生的多个不同的版本,即可构成一个全态数据集。在一个全态数据集中,有的数据项处于当前态,有的数据项处于过渡态,有的数据项处于历史态数据。这里的全态数据集是指一个抽象的、虚拟的集合概念,同一个全态数据集内的各个数据项可以分布式地存储在不同的物理机上。数据库系统在存储各个数据项时,可以采用指针将对应于同一主键标识的各个数据项按照时序链接起来,便于查询全态数据的生命周期轨迹。

[0070] 数据项的可见性:数据项的可见与否(数据项的可见性)是针对于事务而言的,某个数据项可能针对一些事务可见,针对一些事务不可见。在本申请实施例中,提出了数据库系统的多级可串行化隔离级别,并在此基础上针对不同的可串行化隔离级别,提供了不尽相同的可见性判断算法,用以在不同的可串行化隔离级别下判断数据项的可见性,具体的可见性判断算法将在后文进行详细说明,这里不做赘述。

[0071] 本申请实施例所涉及的数据库系统,可以是一种分布式的数据库系统,也可以是一种分布式的大数据处理系统,在分布式系统中可以包括至少一个节点设备,每个节点设备的数据库中存储有数据表,每个数据表可以用于存储一个或多个数据项(也称为元组)。其中,节点设备的数据库可以为任一类型的分布式数据库,可以包括关系型数据库或者非关系型数据库中至少一项,例如SQL(Structured Query Language,结构化查询语言)数据库、NoSQL(Non-relational SQL,泛指非关系型数据库)、NewSQL(泛指各种新式的可拓展/高性能数据库)等各种数据处理系统,在本申请实施例中对数据库的类型不作具体限定。

[0072] 从逻辑的角度出发,可以将分布式系统中节点设备划分为两种角色:协调节点设备(Host Node,也称为计算节点设备)和数据节点设备(Resource Manager,RM),其中,协调节点设备主要负责生成、分发查询计划(也即分发事务的执行请求),以及协调分布式事务,而数据节点设备则主要负责对数据进行分片存放,接收协调节点设备发来的查询计划,执行相应的事务并向协调节点设备返回事务涉及的数据项。

[0073] 在分布式数据库系统中,最小的操作执行单元为事务,依据事务是否需要多个数据节点设备上的数据项进行操作,事务可以被划分为全局事务(又称分布式事务)和本地事务两种,针对这两种不同的事务,可以分别采取不同的执行流程,以尽量减少网络通信开销,提升事务处理效率。其中,全局事务表示事务需要跨多个数据节点设备执行读写操作,也即事务需要对多个数据节点设备上的数据项进行操作,例如,事务T需要操作数据节点设备RM1、RM2、RM3上的数据项,那么该事务T为一个全局事务;本地事务表示事务只需要对单个数据节点设备上的数据项进行操作,例如,事务T只需要操作RM1上的数据项,则该事务T为一个本地事务。

[0074] 在一些实施例中,本申请实施例还可以应用于一种基于区块链技术的数据库系统(以下简称为“区块链系统”),上述区块链系统在本质上属于一种去中心化式的分布式数据库系统,采用共识算法保持区块链上不同节点设备所记载的账本数据一致,通过密码算法保证不同节点设备之间账本数据的加密传送以及不可篡改,通过脚本系统来拓展账本功能,通过网络路由来进行不同节点设备之间的相互连接。

[0075] 在区块链系统中可以包括一条或多条区块链,区块链是一串使用密码学方法相关联产生的数据块,每一个数据块中包含了一批网络交易的信息,用于验证其信息的有效性(防伪)和生成下一个区块。

[0076] 区块链系统中节点设备之间可以组成点对点(Peer To Peer,P2P)网络,P2P协议是一个运行在传输控制协议(Transmission Control Protocol,TCP)协议之上的应用层协议。在区块链系统中,任一节点设备可以具备如下功能:1)路由,节点设备具有的基本功能,用于支持节点设备之间的通信;2)应用,用于部署在区块链中,根据实际业务需求而实现特定业务,记录实现功能相关的数据形成账本数据,在账本数据中携带数字签名以表示数据来源,将账本数据发送至区块链系统中的其他节点设备,供其他节点设备在验证账本数据来源以及完整性成功时,将账本数据添加至临时区块中,其中,应用实现的业务可以包括钱包、共享账本、智能合约等;3)区块链,包括一系列按照先后的时间顺序相互接续的区块,新区块一旦加入到区块链中就不会再被移除,区块中记录了区块链系统中节点设备提交的账本数据。

[0077] 在一些实施例中,每个区块中可以包括本区块存储交易记录的哈希值(本区块的哈希值)以及前一区块的哈希值,各区块通过哈希值连接形成区块链,另,区块中还可以包括有区块生成时的时间戳等信息,比如还可以包括本申请实施例提供的事务的全局状态信息。

[0078] 在分布式系统中,分布式事务处理是一项值得关注的焦点,可以划分为中心化的事务处理技术和去中心化的事务处理技术,下面进行介绍:

[0079] 中心化的事务处理技术:是指分布式数据库中存在某一节点设备为事务管理器,集中对事务进行并发控制,在该节点设备上维护全局事务状态、全局快照等信息,并对系统中所有事务进行统一管理。Postgres-XC(Postgres-eXtensible Cluster,可扩展集群数据库服务器)是采用中心化的事务处理技术的典型例子,在Postgres-XC的架构中,系统内包括一个中心事务管理器,来对全局事务进行管理;系统内还包括多个控制节点(Coordinator,相当于协调节点),来负责SQL的解析和执行;此外,系统内还包括多个数据节点(Data Node),来用于为控制节点提供数据服务。

[0080] 去中心化的事务处理技术:是指分布式数据库中不存在某一节点设备集中对事务进行协调,而是在分布式数据库中多个节点设备能够用于充当事务协调者的角色,由于涉及到多个协调节点设备共同处理分布式事务,因此需要重点保证分布式事务的正确性。目前的主流做法是在多个协调节点设备之间进行通信,并通过特定方法同步全局事务信息,从而保证事务的全局正确性(也称为全局事务一致性)。在采用去中心化的事务处理技术的系统中,比较典型的例子是Spanner系统,在Spanner系统中采用Truetime机制并结合锁机制,实现了去中心化的事务管理,其中,Truetime机制是一种依赖于物理设备(比如GPS和原子钟,其中,GPS指Global Positioning System,全球定位系统)的时间戳排序机制。

[0081] 下面分别针对上述两种不同的分布式事务处理技术进行分析:

[0082] 对中心化的事务处理技术而言,由于事务并发访问控制本身的复杂性,需要耗费较多的系统资源,很容易称为分布式系统的性能瓶颈,而在中心化的事务处理技术中,使用了全局唯一的全局事务管理节点来对所有事务进行管理,那么很容易造成分布式系统中的单点瓶颈问题,使得分布式系统具备的可扩展能力较差,这里所涉及的可扩展性能较差是指:分布式系统的整体性能很容易因为单点的事务管理设计而无法随着机器数量的增加而线性增加,因此基于中心化的事务处理技术的分布式数据库产品,很难应用于大规模的交易场景中,对业务的局限性较大。

[0083] 对去中心化的事务处理技术而言,由于事务的并发控制普遍依赖于锁机制和时间戳排序机制,这两个机制在主流的互联网应用场景(指读操作较多且写操作较少)中会具有较差的性能,导致事务的吞吐量无法得到提升,进一步地,仍然需要多个协调节点设备之间进行相应的全局状态同步,比如需要依赖于一个全局逻辑时钟(这又成为了一个单点),因此也很容易造成数据库系统的单点瓶颈问题,导致数据库系统的可扩展性较差,此外,如果不依赖于全局逻辑时钟,但为了确保事务一致性和分布式系统的一致性,那么整个分布式系统会存在较大的时延(比如Spanner系统采用的Truetime机制就会导致较大时延),因此基于去中心化的事务处理技术的分布式数据库产品的性能有待提升。

[0084] 有鉴于此,在本申请实施例中提供一种事务处理方法,是一种能够适用于分布式数据库系统的新型事务处理机制,首先,创新性的提出了分布式事务的多级可串行化隔离级别,丰富了分布式数据库系统中的可串行化的定义,其次,提出了一套分布式事务处理方案,使得系统具备支持多级可串行化隔离级别的能力,最后,提出了一系列的优化方法,能够提升分布式事务处理的吞吐量、减少回滚率,将在下文进行详述。

[0085] 图1是本申请实施例提供的一种事务处理方法的实施环境示意图。参见图1,本实施例可以应用于分布式数据库系统,该系统中可以包括网关服务器101、全局时间戳生成集群102、分布式存储集群103以及分布式协调系统104(例如ZooKeeper),在分布式存储集群103中可以包括数据节点设备和协调节点设备。

[0086] 其中,网关服务器101用于接收外部的读写请求,并将读写请求对应的读写事务分发至分布式存储集群103,比如,用户在登录终端上的应用客户端之后,触发应用客户端生成读写请求,调用分布式数据库系统提供的API(Application Programming Interface,应用程序编程接口)将该读写请求发送至网关服务器101,比如,该API可以是MySQL API(一种关系型数据库系统提供的API)。

[0087] 在一些实施例中,该网关服务器101可以与分布式存储集群103中的任一个数据节点设备或任一协调节点设备合并并在同一个物理机上,也即是,让某个数据节点设备或协调节点设备充当网关服务器101。

[0088] 全局时间戳生成集群102用于生成全局事务的全局提交时间戳(Global Timestamp,Gts),该全局事务可以是指涉及到多个数据节点设备的事务,例如全局读事务可以涉及到对多个数据节点设备上存储数据的读取,又例如,全局写事务可以涉及到在多个数据节点设备上写入数据。全局时间戳生成集群102在逻辑上可以视为一个单点,但在一些实施例中可以通过一主多从的架构来提供具有更高可用性的服务,采用集群的形式来实现该全局提交时间戳的生成,可以防止单点故障,也就规避了单点瓶颈问题。

[0089] 可选地,全局提交时间戳是一个在分布式数据库系统中全局唯一且单调递增的时间戳标识,能够用于标志每个事务全局提交(提交是一个事件)的顺序,以此来反映出事务之间在真实时间上的先后关系(事务的全序关系),全局提交时间戳可以采用全局逻辑时钟、全局物理时钟、全局混合逻辑时钟(Hybrid Logical Clock,HLC)或者分布式HLC中至少一项,本申请实施例不对全局提交时间戳的类型进行具体限定。

[0090] 在一个示例性场景中,全局提交时间戳可以采用全局逻辑时钟的方式,全局提交时间戳可以由8字节组成,其中,前44位可以为物理时间戳的取值(也即Unix时间戳,精确到毫秒),这样共计可以表示 2^{44} 个无符号整数,因此理论上一共可以表示约为557.8

$(\frac{2^{44}}{1000*60*60*24*365} = 557.8)$ 年的物理时间戳,其中,后20位可以为在某一毫秒内的单调递增计数,这样每毫秒有 2^{20} 个(约100万个)计数,基于上述数据结构,如果单机(任一数据节点设备)的事务吞吐量为10w/s,理论上可以支持包含1万个节点设备的分布式存储集群103,同时,全局提交时间戳的数量代表了系统理论上所能支持的总事务数,基于上述数据结构,理论上系统可以支持 $(2^{44}-1)*2^{20}$ 个事务。这里仅仅是对一种全局提交时间戳的定义方法的示例性说明,根据业务需求的不同,可以对全局提交时间戳的位数进行扩展,以满足对更多的节点数、事务处理数的支持,本申请实施例不对全局提交时间戳的定义方法进行具体限定。

[0091] 在一些实施例中,该全局时间戳生成集群102可以是物理独立的,也可以和分布式协调系统104(例如ZooKeeper)合并到一起。

[0092] 其中,分布式存储集群103可以包括数据节点设备和协调节点设备,每个协调节点设备可以对应于至少一个数据节点设备,数据节点设备与协调节点设备的划分是针对不同事务而言的,以某一全局事务为例,全局事务的发起节点可以称为协调节点设备,全局事务所涉及的其他节点设备称为数据节点设备,数据节点设备或协调节点设备的数量可以是一个或多个,本申请实施例不对分布式存储集群103中数据节点设备或协调节点设备的数量进行具体限定。由于本实施例所提供的分布式数据库系统中缺乏全局事务管理器,因此在该系统中可以采用XA(eXtended Architecture,X/Open组织分布式事务规范)/2PC(Two-Phase Commit,二阶段提交)技术来支持跨节点的事务(全局事务),保证跨节点写操作时数据的原子性和一致性,此时,协调节点设备用于充当2PC算法中的协调者,而该协调节点设备所对应的各个数据节点设备用于充当2PC算法中的参与者。

[0093] 可选地,每个数据节点设备或协调节点设备可以是单机设备,也可以采用主备结

构(也即是为了一主多备集群),如图1所示,以节点设备(数据节点设备或协调节点设备)为一主两备集群为例进行示意,每个节点设备中包括一个主机和两个备机,可选地,每个主机或备机都对应配置有代理(agent)设备,代理设备可以与主机或备机是物理独立的,当然,代理设备还可以作为主机或备机上的一个代理模块,以节点设备1为例,节点设备1包括一个主数据库及代理设备(主database+agent,简称主DB+agent),此外还包括两备数据库及代理设备(备database+agent,简称备DB+agent)。

[0094] 在一个示例性场景中,每个节点设备所对应的主机或备机的数据库实例集合称为一个SET(集合),例如,假设某一节点设备为单机设备,那么该节点设备的SET仅为该单机设备的数据库实例,假设某一节点设备为一主两备集群,那么该节点设备的SET为主机数据库实例以及两个备机数据库实例的集合,此时可以基于云数据库的强同步技术来保证主机的数据与备机的副本数据之间的一致性,可选地,每个SET可以进行线性扩容,以应付大数据场景下的业务处理需求,在一些金融业务场景下,全局事务通常是指跨SET的转账。

[0095] 分布式协调系统104可以用于对网关服务器101、全局时间戳生成集群102或者分布式存储集群103中至少一项进行管理,可选地,技术人员可以通过终端上的调度器(scheduler)访问该分布式协调系统104,从而基于前端的调度器来控制后端的分布式协调系统104,实现对各个集群或服务器的管理。例如,技术人员可以通过调度器来控制ZooKeeper将某一个节点设备从分布式存储集群103中删除,也即是使得某一个节点设备失效。

[0096] 上述图1仅是提供了一种轻量级的全局事务处理的架构图,是一种类分布式数据库系统。整个分布式数据库系统可以看作是共同维护一个逻辑上的大表,这个大表中存储的数据通过主键被打散到分布式存储集群103中的各个节点设备中,每个节点设备上存储的数据是独立于其他节点设备的,从而实现了节点设备对逻辑大表的水平切分。由于在上述系统中能够将各个数据库中各个数据表水平切分后进行分布式地存储,因此,这种系统也可以形象地称为具有“分库分表”的架构。

[0097] 在上述分布式数据库系统中,已经基于XA/2PC算法实现了写操作时数据的原子性和一致性,而读操作的数据一致性问题,需要通过构造一个轻量的、去中心化的分布式事务处理机制来改善,从技术的角度来看,分布分表架构缺乏一个全局事务管理器,也就缺乏分布式事务处理能力,通过构造上述轻量的、去中心化的分布式事务处理机制,能够为分布式数据库系统提供水平扩展等能力,并且保证分布式数据库系统简单易推广、事务处理效率更高,必将对传统并发控制方式所设计的分布式数据库架构产生极大冲击,具体的分布式事务处理机制将在下个实施例中进行详述。

[0098] 在一些实施例中,上述网关服务器101、全局时间戳生成集群102、分布式存储集群103以及分布式协调系统104所构成的分布式数据库系统,可以视为一种向用户终端提供数据服务的服务器,该服务器可以是独立的物理服务器,也可以是多个物理服务器构成的服务器集群或者分布式系统,还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN(Content Delivery Network,内容分发网络)、以及大数据和人工智能平台等基础云计算服务的云服务器。可选地,上述用户终端可以是智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱、智能手表等,但并不局限于此。终端以及服务器可以通过有线或无线通信方式进行直接或间接地连接,本申请在

此不做限制。

[0099] 在介绍本申请实施例之前,由于事务并发控制的正确程度可以通过一致性和隔离性来描述,下面对一致性和隔离性进行解释说明:

[0100] 一、隔离性

[0101] 事务隔离级别通过能否规避某种数据异常而进行定义,可能涉及到的数据异常包括:1)脏读,指一个事务读取到了另一事务尚未提交的数据项;2)不可重复读,指一个事务对同一数据项重复读取两次却得到了不同的结果;3)幻读,指事务在操作过程中进行两次范围查询,第二次查询的结果包含了第一次查询的结果中未出现的数据项或者缺少了第一次查询的结果中出现的数据项。

[0102] 基于能够解决上述三种数据异常,标准SQL中定义了四种隔离级别,分别包括:1)读未提交:允许如上三种数据异常发生;2)读已提交:不允许脏读发生;3)可重复读:不允许脏读、不可重复读发生;4)可串行化:如上三种数据异常均不能发生。

[0103] 另外,还需要注意的一种数据异常称为丢失更新异常,是指两个并发事务同时进行更新,而后一个事务的更新覆盖了前一个事务的更新的情况,丢失更新异常是由于数据没有保证一致性而导致的。比如,存在数据项 r_1 , r_1 中记录了属性值 $x=100$, t 时刻下事务 w_1 和 w_2 同时对数据项 r_1 进行更新,事务 w_1 首先将 $x=100$ 修改为 $x=120$ 之后提交,随后事务 w_2 又将 $x=100$ 修改为 $x=130$ 之后提交,导致在查询事务 w_1 时,会发现刚才修改的内容没有被修改掉,就好像是“丢失了更新”,因此这种数据异常称为丢失更新异常,丢失更新异常在可重复读和可串行化的一致性级别下均不允许发生。

[0104] 二、一致性

[0105] 首先,数据库的一致性定义为:在事务的操作下,数据库的数据状态从一个一致的状态变迁为另一个一致的状态。上述“一致的状态”是指满足数据库系统预先定义的一些规则的数据状态,比如,这些规则可以包括约束、级联、触发器以及三者之间任意的组合(属于数据的逻辑语义),写偏序异常违反的就是特定数据之间的约束,这里的约束属于用户语义所限定的数据的一致性。

[0106] 对于整个数据库系统而言,一致性还包括一层系统级的含义,是指要想保证数据在数据库系统中保持一致,还要求数据库系统符合两个特性,一个是可串行性(serializability),另一个是可恢复性(recoverability)。可串行性也即是说上述在隔离性中所定义到的可串行化隔离级别,可串行性保证了数据不会被并发操作改坏,而可恢复性是指已经提交的事务未曾读过被回滚的事务写过的数据(指不会发生脏读异常),可恢复性保证了事务被回滚后数据回到之前的一致状态,被回滚的事务不会对数据的一致性造成影响,数据库系统的一致性状态是可恢复的。

[0107] 再者,对分布式系统尤其是分布式数据库系统而言,需要考虑分布式系统层面的一致性。分布式系统层面的一致性用来描述事务之间的操作顺序是否符合某种约束,换言之,通过对事务间原有的先后关系的定义进行细化,从而保留已经确定的先后关系约束,主要可以划分为:

[0108] 1) 线性一致性,保证当前操作序列与一个保留了全部原有符合真实时间的先后关系 $<_H$ 的顺序历史等效,即要求所有操作之间存在全序关系,并且需要保留全部的真实时间先后顺序,其中,先后关系 $<_H$ 是指,假设存在操作A和操作B,如果操作A的结束时刻处于操

作B的开始时刻之前,那么操作A和操作B之间的全序关系可以记为“ $A <_H B$ ”。

[0109] 2) 顺序一致性,保证当前操作序列的执行结果与某一顺序序列等价,即要求所有操作之间存在全序关系,而不用保留符合真实时间的先后关系 $<_H$ 。

[0110] 由于线性一致性和顺序一致性这两种顺序均是构建在全局顺序范围内的,基于是否符合全局真实时间(RealTime)先后顺序进行区分,因此能够与全局的事务的一致性(即可串行化)进行结合,以构造出多种分布式事务一致性级别。在本申请实施例中,聚焦于可串行化隔离级别,定义了五种可串行化隔离级别,下面将分别进行介绍。

[0111] 为了更加清楚的说明五种可串行化隔离级别,首先需要引入事务串行调度的概念,事务串行调度的含义是在某一时刻,系统内只允许至多一个事务执行。而在数据库系统中,事务是允许并发和并行执行的。并发和并行的区别在于,如果两个事务操作了同一数据项,这两个事务被称作并发事务;如果两个事务没有操作同一数据项,这两个事务被称为并行事务。

[0112] 在事务的调度中,正确的法则是做到可串行化。如果存在调度S,对于数据库的任何状态,其执行结果完全等价于另一个串行调度S',则称这样的调度S为可串行化调度。如果只遵循传统的可串行化定义,调度S中存在的事务间的原有先后关系,在其等价的串行调度S'中。

[0113] 再者,在提供了多级可串行化隔离级别的基础上,还需要引入事务的生命周期这一概念,事务的生命周期可以被划分为:逻辑执行生命周期和实际执行生命周期。

[0114] 逻辑执行生命周期:代表事务在排定的串行序列中的生命周期。

[0115] 实际执行生命周期:代表事务实际并发或并行执行时,所具有的生命周期。

[0116] 结果生命周期:在上述两种生命周期的基础上,最终,在一个事务结束后,会形成一个最终生命周期,称为事务的结果生命周期。需要说明的是,回滚事务无结果生命周期,只有提交完毕的事务才能在数据上留下事务一致性的痕迹,也即是只有提交完毕的事务才具有结果生命周期。此结果生命周期,构成了事务操作在数据项上的一致性状态,从而保障后发的事务提供正确的一致性读;结果生命周期一旦确定,即结果存在唯一性,无歧义。

[0117] 在本申请实施例中,通过研究分布式事务处理中的可串行化问题,在一个系统的全局范围内定义了多级可串行化隔离级别,从而在可串行化的基础上,通过不同的级别对事务原有的先后关系进行不同的约束,能够为分布式系统统一了事务先后关系的衡量标准,下面将结合事务的生命周期来分析不同的可串行化隔离级别,上述多级可串行化隔离级别分别包括:

[0118] 1、线性可串行化(Linearizable Serializability,LS):所有事务之间如果存在符合真实时间的先后关系 $<_H$,在等价的事务顺序执行序列中,其真实时间先后关系 $<_H$ 被保留。即LS隔离级别中所有事务(并发事务+并行事务),均保持全序。

[0119] 换言之,对于任意两个实际执行生命周期不相交的事务,这两个事务的逻辑执行生命周期需要体现原有的实际执行顺序。每个事务的结果生命周期由逻辑执行生命周期和实际执行生命周期共同决定。

[0120] 2、线性并发结果可串行化(Linearizable Concurrent Result Serializability,LCR):并发事务之间如果存在符合真实时间的先后关系 $<_H$,在等价的事务顺序执行序列中,其真实时间先后关系 $<_H$ 被保留。即LCR隔离级别中并发事务按可串行

化顺序,保持全序;并行事务按提交顺序,在全局系统范围内保持全序。

[0121] 换言之,对于任意两个实际执行生命周期相交的并行事务,这两个事务的实际执行生命周期取决于各自的提交顺序。每个事务的结果生命周期由实际执行生命周期决定。

[0122] 3、线性点可串行化 (Linearizable Point Serializability, LP):在概念上与LCR隔离级别相同,但具体要求存在不同。即LP隔离级别中,仍然是并发事务按可串行化顺序,保持全序;并行事务按提交顺序,在全局系统范围内保持全序。但区别在于,LP隔离级别中每个事务的开始时间和结束时间,在同一个RealTime点上;而LS隔离级别和LCR隔离级别的事务生命周期,可以是一个线段。

[0123] 换言之,对于任意两个实际执行生命周期可能相交的并发事务,这两个事务的逻辑执行生命周期要决定原有的实际执行顺序,使得最终的事务生命周期(结果生命周期)蜕化为一个点值。事务间的先后关系通过时间点形式体现。

[0124] 以上3种LS、LCR、LP隔离级别,对于并发事务而言,均需要用事务的可串行化顺序(逻辑执行生命周期)确定他们全局顺序(实际执行生命周期)。

[0125] 4、顺序可串行化 (Sequential Result Serializability, SR):并发事务之间保证可串行化,并且所有事务(并发事务+并行事务)之间存在全序关系,即所有事务执行结果与某一确定的顺序执行序列等价。即在可串行化的基础上,强调事务的结果是稳定的,对于后发生的事务所能读到的数据状态处于稳定不变。

[0126] 换言之,对于任意两个事务,这两个事务的实际执行生命周期能够确定这两个事务之间唯一的先后关系。

[0127] 5、顺序并发结果可串行化 (Sequential Concurrent Result Serializability, SCR):并发事务之间保证可串行化,即并发事务执行结果与某一确定的顺序执行序列等价,并发事务的事务序确定了并发事务的全局序,而并行事务,不存在全局序。

[0128] 换言之,对于任意两个并发事务,这两个事务的逻辑执行生命周期能够确定这两个事务之间唯一的先后关系,不考虑事务的实际执行生命周期。

[0129] 在上述过程中,针对分布式事务提出了多级可串行化隔离级别,丰富了数据库系统中可串行化隔离级别的定义,从而为分布式事务的正确性提供了衡量标准,通过采用不同级别的可串行化,分布式系统可以在可串行这个正确性前提下,提供多种不同的可串行化实现机制,从而提供可调节的可串行化事务处理性能。

[0130] 基于上述多级可串行化隔离级别的数据库系统,能够适用于多种业务场景,如果采用严格的可串行化隔离级别(比如LS隔离级别),能够很好地适用于金融领域,以严格保证数据的可靠性,而传统的分布式数据库系统均无法高效地提供这一可串行化隔离级别,同时,如果采用较弱的可串行化隔离级别(比如SCR隔离级别),可以很好地适用于互联网场景,能够提供高并发、实时的数据库服务,为互联网用户提供良好的产品体验,能够提升基于数据库系统的产品竞争力,具有较高的技术影响力和现实意义。

[0131] 以下,将结合具体的示例来阐述不同的可串行化隔离级别之间的区别与联系。在下述示例中,采用T代表事务,采用W代表写操作,采用R代表读操作,采用C代表提交操作,采用P代表某一进程,每一个事务的范围则代表该事务的实际执行生命周期。

[0132] 1、线性可串行化LS与线性并发结果可串行化LCR之间本质区别在于:线性+可串行化的LS隔离级别下,并行事务之间也需要保留原有的实际执行先后关系;而并发线性+可串

行化的LCR隔离级别下,则不需要保留并行事务之间的实际执行先后关系。

[0133] 图2是本申请实施例提供的一种事务执行顺序的原理性示意图,如200所示,进程P1用于执行事务T1,事务T1写入了数据项x1后提交,记为“W1 (x1) C1”,进程P2用于执行事务T2,事务T2写入了数据项y1后提交,记为“W2 (y1) C2”,且事务T1在事务T2实际开始之前,就已经提交。线性+可串行化的LS隔离级别下,对于事务T1和事务T2来说,事务T1的逻辑执行生命周期一定在前,事务T2的逻辑执行生命周期一定在后,即对并行事务也只允许1种调度方式:T1→T2;而并发线性+可串行化的LCR隔离级别下,可以不用体现并行事务之间的这一先后关系,即允许2种调度方式:T1→T2或者T2→T1,因为LCR隔离级别中有多种调度可能,这便为存在级联关系的事务进行合理调度留下了调优空间。

[0134] 2、线性并发结果可串行化LCR与顺序可串行化SR之间本质区别在于:并发线性+可串行化的LCR隔离级别下,事务逻辑执行生命周期逻辑需要体现并发事务间原有的实际执行先后关系,而顺序可串行化SR隔离级别下,不需要体现。

[0135] 图3是本申请实施例提供的一种事务执行顺序的原理性示意图,如300所示,进程P1用于执行事务T2,事务T2读取数据项x1、写入数据项x2、读取数据项y2后提交,其提交顺序为第三位,记为“R2 (x1) W2 (x2) R2 (y1) C3”,进程P2用于执行事务T1,事务T1写入数据项x1和数据项y1后提交,其提交顺序为第一位,记为“W1 (x1) W1 (y1) C1”,进程P3用于执行事务T3,事务T3写入数据项y2后提交,其提交顺序为第二位,记为“W3 (y2) C2”,进程P4用于执行事务T4,事务T4读取数据项x1后提交,其提交顺序为第四位,记为“R2 (x1) C4”。可以看出,事务T3和事务T4之间存在原有实际执行顺序为T3在前、T4在后的先后关系。而对于事务T3和事务T4的逻辑执行生命周期,在本例中以逻辑执行生命周期T4在前、T3在后为例进行说明,此时逻辑执行生命周期不能体现原有的实际执行顺序。因此,在线性并发结果可串行化LCR隔离级别下,本例中的部分事务执行不允许;而在顺序可串行化SR隔离级别下,本例中事务均允许执行。

[0136] 3、顺序可串行化SR与顺序并发结果可串行化SCR之间本质区别在于:顺序可串行化SR隔离级别下要体现并行事务之间的顺序;而顺序并发结果可串行化SCR隔离级别下只体现并发事务之间的顺序。

[0137] 如图2所示,在顺序可串行化SR隔离级别下,事务T1和事务T2之间的顺序应该被唯一确定,而在顺序并发结果可串行化SCR隔离级别下,这两个事务间的顺序不用被唯一确定。

[0138] 在提供了多级可串行化隔离级别的基础上,在此对本申请实施例所涉及的基本数据结构进行解释说明:

[0139] 一、数据项结构

[0140] 本申请实施例所涉及的数据项结构(也称为数据版本结构)可以适用于段页式存储的数据库系统或者键值式(key-value)存储的数据库系统中至少一项,由于段页式存储的数据结构可以基于键值式存储的数据结构获取,因此,在本申请实施例中以键值式存储的数据结构为例进行说明,数据项结构可以如图4所示,对任一数据项400(也称为元组、数据版本)来说,数据项的键401(Key)可以为<User_key,Lts,Gts>,数据项的值402(Value或者Data)可以为其余属性值。

[0141] 其中,User_key是由用户定义的主键,在数据库系统中默认用户需要为数据表定

义主键。

[0142] 其中,Lts是指产生该数据项的事务的逻辑提交时间戳,也即是写入该数据版本的事务的逻辑提交时间戳,在事务提交之后赋值,这里涉及的逻辑提交时间戳用于表示事务的逻辑提交时刻。

[0143] 其中,Gts是指产生该数据项的事务的全局提交时间戳,也即是写入该数据版本的事务的全局提交时间戳,在事务提交之后赋值,这里涉及的全局提交时间戳用于表示事务的全局提交时刻,可以由上述实施环境中的全局时间戳生成集群来进行分发。

[0144] 需要说明的是,由于全局提交时间戳包括多种数据类型,比如全局逻辑时钟、全局物理时钟、全局HLC、分布式HLC等,随着全局提交时间戳的数据类型发生变化,逻辑提交时间戳的数据类型也将随之变化,保证与全局提交时间戳相关的其他时间戳结构的数据类型一致,比如,假设全局提交时间戳格式被定义为全局HLC,那么相应地,全局逻辑提交时间戳也要采取全局HLC的格式,后文中将不做赘述。

[0145] 在一些实施例中,对于上述事务结构中所提到的逻辑提交时间戳Lts,如上所言,可以采用两种实现方式:采用全局LC即全局逻辑时钟进行实现,可以保证Lts的正确性;或,采用全局或分布式HLC即全局或分布式混合逻辑时钟进行实现,可以在执行效率上得到优化。

[0146] 在一个示例性场景中,以逻辑提交时间戳Lts采用全局HLC为例进行说明,HLC是一个由相关专业人员提出的分布式的时间戳获取算法,从逻辑时钟基础上演化而来。HLC在保证因果序的前提下,又可以和物理时钟对应起来。HLC时间戳分为两个部分:WallTime+LogicTime。

[0147] WallTime部分:实际上代表了当前节点已知的最大物理时间(当前节点和与当前节点有因果关系节点的最大物理时间)。

[0148] LogicTime部分:如果几个节点上WallTime一样,用LogicTime部分来区分事件发生的先后关系。

[0149] 在给本地节点产生的事件分配HLC时间时,WallTime部分取当前WallTime和当前物理时间之间的最大值。如果物理时间小于或等于WallTime,LogicTime在原有基础上加一;如果物理时间大于WallTime,LogicTime归零。

[0150] 二、数据项页眉结构

[0151] 一个数据项集合(也即全态数据集)可以由多个数据版本和一个数据项页眉结构(Header)构成,简而言之,针对具有相同主键标识(User_key)的各个数据项而言,可以维护同一个页眉结构,页眉结构中至少可以存储下述数值:

[0152] 1) User_key,用户定义主键,与各个数据版本的键中存储的User_key相同。

[0153] 2) Rts,记录读取过该数据项的所有事务的逻辑提交时间戳中的最大值,也可以称为最大读事务时间戳。

[0154] 3) WT,表示该数据项所对应的待写事务,WT中可以记录要写入该数据项的事务的事务标识(TID)。

[0155] 4) RTlist,记录访问过该数据项集合中最新数据版本的活跃事务集合,也可以称为读事务列表,该活跃事务集合可以是数组的形式,也可以是列表、队列、堆栈等形式,本申请实施例不对活跃事务集合的形式进行具体限定,RTlist中每个元素可以是读取过上述最

新数据版本的事务的事务标识(TID)。

[0156] 三、事务读集结构

[0157] 任一个事务的读集结构中记录了该事务所涉及读取的数据项,可以使用内存链表结构来维护事务的读集。需要说明的是,对一个全局读事务而言,其读集可以划分为本地读集和全局读集,本地读集存在于数据节点设备RM上,而全局读集存在于协调节点设备上,当然,协调节点设备可以定期将全局读集同步至各个数据节点设备上,使得数据节点设备上也能够维护事务的全局读集。

[0158] 在基于内存链表结构维护事务读集的基础上,每个链表节点可以对应于一个读取到的数据版本的key值,key值可以包括下述两个属性:1) Size,取4字节,用于表示key所占字节数;2) Key,在Size之后的一个可以变长的字段,记录了读取到的数据版本的key值。

[0159] 在一些实施例中,可以提供一种基于范围的读集结构优化策略,由于存在范围查询等需要一次性读取较多数据的情况,因此在上述事务读集结构的基础上,可以提供一种基于范围的读集维护优化方法,减少事务读集结构的维护开销。可选地,基于范围的读集结构仍然可以采用链表结构,链表中每个节点包含四个属性:1) Start-Key-Size,取4字节,表示Start-Key所占字节数;2) Start-Key,在Start-Key-Size之后的一个可以变长的字段,代表查询范围的开始处的主键值;3) End-Key-Size,取4字节,表示End-Key所占字节数;4) End-Key,在End-Key-Size之后的一个可以变长的字段,代表查询范围的结束处的主键值。

[0160] 其中,Start-key和End-key的确定,可以由查询解析器完成,通过查询时给出的谓词条件转化而来。

[0161] 四、事务写集结构

[0162] 任一个事务的写集结构中记录了该事务需要更新的数据项,与读集结构类似,同样可以使用内存链表结构来维护事务的写集。需要说明的是,对一个全局写事务而言,其写集可以划分为本地写集和全局写集,本地写集存在于数据节点设备RM上,而全局写集存在于协调节点设备上,当然,协调节点设备可以定期将全局写集同步至各个数据节点设备上,使得数据节点设备上也能够维护事务的全局写集。

[0163] 在基于内存链表结构维护事务写集的基础上,每个链表节点可以对应于一个写集中的数据项,所记录的数据项可以包括下述两个属性:1) Size,取4字节,用于表示数据项的大小,也即表示Version属性所占字节数;2) Version,在Size之后的一个可以变长的字段,用于表示数据项的key-value键值,记录了需要插入/更新的数据项(数据版本)。

[0164] 五、事务的全局状态信息

[0165] 对于任一个事务T,该事务的全局状态信息可以表示为一个{TID,Lowts,Uppts,Lts,Gts,SI,Status}形式的七元组,该七元组也可以称为事务T的全局事务状态、事务状态结构等,该全局状态信息可以同时存在于数据节点设备和协调节点设备上。

[0166] 其中,TID为事务标识,是一个全局唯一的事务编号。

[0167] 在一个示例性场景中,TID可以由8字节组成,前14位用来记录处理该事务的协调节点设备的编号。14位共可以表示16384 (2^{14}) 个无符号整数,因此,可以和估算得到的全局提交时间戳Gts所能支持的节点数相对应。后50位由该协调节点设备内的单调递增计数填充,上述单调递增计数用于区分协调节点设备中的不同事务(共 2^{50} 个),该TID的数量级理论上可保证TID在全局提交时间戳Gts所规定的总事务数范围内不会重复。

[0168] 其中,基于Lowts和Uppts可以确定事务的逻辑执行生命周期,将Lowts确定为逻辑执行生命周期的时间戳下界,将Uppts确定为逻辑执行生命周期的时间戳上界,那么该事务的逻辑执行生命周期可以表示为:[Lowts,Uppts)。事务的逻辑执行生命周期是相对的,在事务执行和验证过程中通常会对其逻辑执行生命周期进行初始化以及后续调整,具体初始化规则和调整规则将在下述实施例中进行详述,这里不做赘述。

[0169] 其中,Lts为事务的逻辑提交时间戳,该逻辑提交时间戳用于表示事务的逻辑提交时刻,在事务提交时从区间[Lowts,Uppts)中获得,通常的获取规则是:Lts=Lowts+1,也就是说,将调整后的逻辑执行生命周期的时间戳下界加一所得的数值确定为事务的逻辑提交时间戳。

[0170] 其中,Gts为事务的全局提交时间戳,该全局提交时间戳用于表示事务的全局提交时刻,在事务提交时可以从上述实施环境中的全局时间戳生成集群中获取当前的全局时间戳作为全局提交时间戳。

[0171] 其中,SI为事务的事务快照时间戳,该事务快照时间戳用于表示事务的全局开始时刻,在事务开始时可以从上述实施环境中的全局时间戳生成集群中获取当前的全局时间戳作为事务快照时间戳,SI是在读取数据时用于判断数据可见性所需要用到的时间戳。

[0172] 其中,Status用于描述事务状态,例如采用1字节大小,任一事务可以具有如下7种状态:正在运行状态(Running)、正在验证状态(Validating)、验证通过状态(Validated)、正在提交状态(Committing)、提交完成状态(Committed)、正在回滚状态(Aborting)和回滚完成状态(Aborted)。

[0173] 在提出上述多级可串行化隔离级别以及基本数据结构的基础上,本申请实施例提供一套分布式事务处理方案,使得分布式数据库系统可以具备同时支持多级可串行化隔离级别的能力,下面对事务的整体执行流程(也即事务执行整体算法)进行详述。图5是本申请实施例提供的一种事务处理方法的交互流程图,参见图5,该实施例包括:

[0174] 501、协调节点设备与终端建立会话,该会话用于处理目标事务。

[0175] 终端可以是用户所对应的任一电子设备,包括但不限于:智能手机、平板电脑、笔记本电脑、台式计算机、智能音箱或者智能手表中至少一项,本申请实施例不对终端的类型进行具体限定。

[0176] 可选地,在终端上可以安装有应用客户端,该应用客户端可以是能够提供数据服务的任一客户端,例如,该应用客户端可以是支付应用客户端、外卖应用客户端、打车应用客户端或者社交应用客户端中至少一项,本申请实施例不对应用客户端的类型进行具体限定。

[0177] 其中,目标事务可以是全局事务,也可以是局部事务,本申请实施例以目标事务为全局事务为例进行说明。

[0178] 在本申请实施例中,仅以协调节点设备(coordinator)为目标事务的发起节点、数据节点设备(participants,或cohort)为目标事务所涉及的参与节点为例进行说明,可选地,除了目标事务的发起节点之外,协调节点设备也可以是上述实施环境中的网关服务器,还可以是分布式存储集群中的任一个节点设备,该数据节点设备可以是目标事务所涉及读写操作的数据项所在的节点设备,还可以是分布式存储集群中的所有节点设备,本申请实施例不对协调节点设备与数据节点设备的数量和类型进行具体限定。

[0179] 需要说明的是,当协调节点设备为目标事务的发起节点时,由于不同的目标事务通常具有不同的发起节点,因此对不同的目标事务而言协调节点设备或者数据节点设备并非固定不变的,也即是说,同一节点设备有可能对一些目标事务而言属于协调节点设备,对另一些目标事务而言属于数据节点设备。

[0180] 在一些实施例中,在会话(Session)建立阶段,终端上的应用客户端可以通过下述方式与数据库系统中某一协调节点设备建立会话:应用客户端发出目标事务T的执行请求,元信息系统(比如上述实施环境中的网关服务器)检查当前客户端是否已经与某一协调节点设备建立会话,如果已建立会话,则复用当前已建立的会话;否则,系统随机选择某一协调节点设备与该应用客户端建立会话关系,该应用客户端之后发出的所有请求均由该协调节点设备负责执行。

[0181] 502、协调节点设备对目标事务进行初始化。

[0182] 在上述事务初始化阶段,协调节点设备对目标事务的事务执行上下文信息进行初始化,具体地,可以执行下述四项初始化操作中的至少一项:

[0183] 1) 协调节点设备为目标事务分配事务标识TID,也即是说,协调节点设备为事务分配全局唯一的事务编号。分配TID的方式可以包括至少一种,例如,采用全局唯一的逻辑时钟的方式生成TID(此时可以从全局时间戳生成集群中获取一个全局逻辑时钟作为TID),又比如,还可以由每个协调节点设备按照下述规则分配TID:将该协调节点设备的节点名作为前缀,将该协调节点设备上维护的一个局部逻辑时钟的值作为后缀,前缀+后缀构成一个全局唯一的事务编号TID,本申请实施例不对分配TID的实现方式进行具体限定。

[0184] 2) 协调节点设备记录目标事务的全局状态信息,其中,由于全局状态信息可以表示为{TID,Lowts,Uppts,Lts,Gts,SI,Status}形式的七元组,TID是由上述初始化操作1)中分配的,此时,可以将Status初始化为正在运行状态Running。

[0185] 3) 协调节点设备对目标事务的逻辑执行生命周期进行初始化。在初始化过程中,各个可串行化隔离级别均可以将逻辑执行生命周期的时间戳上界Uppts统一初始化为 $+\infty$,但在不同的可串行化隔离级别中,对逻辑执行生命周期的时间戳下界具有不同的初始化规则:

[0186] 在线性可串行化LS、线性并发结果可串行化LCR或者线性点可串行化LP中任一隔离级别下,协调节点设备与全局时间戳生成集群进行通信,获取一个当前全局时间戳,并将获取到的当前全局时间戳赋值给逻辑执行生命周期的时间戳下界Lowts。

[0187] 在顺序可串行化SR或者顺序并发结果可串行化SCR中任一隔离级别下,协调节点设备无需与全局提交时间戳通信,而是从自身设备的时间戳机制中获取当前时间,将获取到的当前时间赋值给逻辑执行生命周期的时间戳下界Lowts。

[0188] 4) 协调节点设备对事务快照时间戳SI进行初始化,将事务快照时间戳SI初始化为逻辑执行生命周期的时间戳下界Lowts,使得SI=Lowts。

[0189] 需要说明的是,由于目标事务尚未提交,而Lts和Gts是在事务提交之后赋值的,因此可以在初始化过程中可以将全局状态信息中的Lts和Gts置为空。

[0190] 503、协调节点设备向数据节点设备发送目标事务的执行请求。

[0191] 在上述过程中,协调节点设备可以基于应用客户端发起的请求,优化SQL并生成目标事务的物理执行计划,将执行计划进行分解,分别发送给目标事务所涉及的数据节点设

备,数据节点设备的数量可以为一个或多个,在本申请实施例中不对数据节点设备的数量进行具体限定。

[0192] 504、数据节点设备响应于该执行请求,执行目标事务,向协调节点设备返回目标事务的执行结果。

[0193] 在上述过程中,数据节点设备根据协调节点设备的执行计划来进行实际的数据读写操作,并将执行结果返回给协调节点设备,具体数据节点设备如何执行目标事务以及在执行之前对逻辑执行生命周期的验证操作将在下个实施例中进行详述,此处不做赘述。

[0194] 505、协调节点设备汇总数据节点设备返回的执行结果,将汇总后的执行结果返回至终端。

[0195] 在上述过程中,由于数据节点设备可以会有一个或多个,因此协调节点设备需要对执行结果进行汇总,汇总完毕后返回至客户端。比如,客户端请求读取10个数据项,这10个数据项中有5个数据项存储在数据节点设备RM1上,其余的5个数据项存储在数据节点设备RM2上,RM1和RM2各自将5个数据项返回给协调节点设备,协调节点设备对数据项进行汇总得到10个数据项,并将这10个数据项返回至客户端。

[0196] 上述步骤503-505可以视为目标事务的事务执行阶段,在事务执行阶段结束后,可以进入下述步骤506中的事务验证阶段。

[0197] 506、协调节点设备向数据节点设备发送目标事务的验证请求。

[0198] 在一些实施例中,若目标事务为全局事务,由于全局事务涉及到跨节点的读写操作,协调节点设备需要向所有相关的数据节点设备发送验证请求。

[0199] 在一些实施例中,若目标事务为本地事务,由于本地事务仅涉及到单个数据节点设备的读写操作,因此协调节点设备仅需要向单个数据节点设备发送验证请求。

[0200] 507、数据节点设备响应于该验证请求,对目标事务进行冲突验证,向协调节点设备返回验证结果。

[0201] 在一些实施例中,若目标事务为全局事务,任一数据节点设备响应于验证请求,对目标事务进行冲突验证,若验证通过,数据节点设备向协调节点设备返回验证通过信息,否则,若验证失败,数据节点设备向协调节点设备返回验证失败信息,上述验证通过信息和验证失败信息统称为验证结果。

[0202] 在一些实施例中,若目标事务为本地事务,单个数据节点设备响应于验证请求,对目标事务进行冲突验证,若验证通过,由于不涉及到对其他的数据节点设备的验证结果进行汇总,因此直接进入目标事务的提交阶段,否则,若验证失败,向协调节点设备发送验证失败信息,上述验证通过信息和验证失败信息统称为验证结果。

[0203] 508、协调节点设备汇总数据节点设备的验证结果,确定目标事务的全局验证结果。

[0204] 在上述过程中,若目标事务是全局事务,协调节点设备汇总各个数据节点设备上报的验证结果之后,若各个验证结果均为验证通过信息,那么将全局验证结果确定为“验证通过”,否则,只要存在任一个验证结果为验证失败信息,那么将全局验证结果确定为“验证不通过”。

[0205] 509、协调节点设备响应于全局验证结果为验证通过,向数据节点设备发送目标事务的提交指令。

[0206] 在上述过程中,若目标事务是全局事务,响应于全局验证结果为验证通过,协调节点设备可以与全局时间戳生成集群进行通信,获取当前时刻的全局时间戳作为目标事务的全局提交时间戳Gts,向所有相关的数据节点设备发送提交指令,由各个数据节点设备执行本地的提交操作。

[0207] 在一些实施例中,若目标事务是本地事务,由于直接由单个数据节点设备提交目标事务,因此无需执行上述步骤507-509中与协调节点设备的一轮通信,而是在单个数据节点设备对目标事务验证通过之后,直接由该数据节点设备与全局时间戳生成集群进行通信,获取当前时刻的全局时间戳作为目标事务的全局提交时间戳Gts,再由该数据节点设备执行本地的提交操作。

[0208] 510、数据节点设备响应于该提交指令,提交目标事务。

[0209] 在上述过程中,数据节点设备需要将目标事务的本地写集中的数据落盘,还涉及到一系列基于多级一致性模型的处理操作,将在后文实施例中进行详述,这里不做展开说明。

[0210] 在上述步骤509-510中,提供了对目标事务全局验证通过时的事务提交流程,而在一些实施例中,协调节点设备响应于全局验证结果为验证失败,可以向数据节点设备发送目标事务的回滚指令,数据节点设备响应于该回滚指令,回滚目标事务,也即是在本地进行目标事务的回滚操作。

[0211] 本申请实施例中介绍了分布式数据库系统中事务的整体执行流程,分别按照目标事务为全局事务和本地事务进行了说明,整体执行流程可以划分为五个阶段,分别是步骤501中的会话建立阶段,步骤502中的事务初始化阶段,步骤503-505中的事务执行阶段,步骤506-508中的事务验证阶段,以及步骤509-510中的事务提交阶段。

[0212] 需要说明的是,在事务执行阶段中,数据节点设备可以根据目标事务所读取到的目标数据项,对目标事务的逻辑执行生命周期进行调整。而在事务验证阶段中,数据节点设备可以根据目标事务所欲写入的待写入数据项,对目标事务的逻辑执行生命周期进行调整。最终,协调节点设备对调整后的逻辑执行生命周期进行校验的过程也即是冲突验证的过程,可以判断出目标事务是进入到事务提交阶段还是事务回滚阶段。

[0213] 在各个不同的事务处理阶段中,通过不同系统模块的组合,可以在分布式数据库系统中实现不同的可串行化隔离级别,在此基础上的分布式事务处理方案,能够支持上述多级可串行化隔离级别各自的事务处理机制,使得系统具备同时支持多级可串行化隔离级别的能力,比如,不同的可串行化隔离级别对应于不尽相同的可见性判断算法,并且在不同的可串行化隔离级别下,针对当前事务的逻辑执行生命周期也具有不同的调整策略,在后续实施例中,将对数据节点设备上的事务执行流程以及事务验证流程分别进行详细说明。

[0214] 图6是本申请实施例提供的一种事务处理方法的流程图,请参考图6,该实施例应用于数据节点设备,本申请实施例中针对上述实施例中事务执行阶段的步骤504中如何执行目标事务以及事务验证阶段的步骤507中如何验证目标事务来进行展开说明,下面进行详述:

[0215] 601、数据节点设备响应于目标事务的执行请求,获取该目标事务的逻辑执行生命周期,该逻辑执行生命周期用于表示该目标事务在事务处理过程中的逻辑时间戳区间。

[0216] 在上述过程中,数据节点设备在接收到协调节点设备发送的目标事务的执行请求

之后,可以获取该目标事务的全局状态信息,由于该全局状态信息中携带逻辑执行生命周期的时间戳上界Uppts以及时间戳下界Lowts,因此可以根据该全局状态信息确定出该逻辑执行生命周期[Lowts,Uppts)。

[0217] 可选地,在获取全局状态信息时,数据节点设备可以根据目标事务T的全局事务标识TID(也即事务编号),在本数据节点设备中查询是否已缓存该目标事务T的全局状态信息,如果没有查询到相应的全局状态信息,则在本数据节点设备中初始化目标事务在当前时刻的全局状态信息,将该全局状态信息赋值为协调节点设备在发送该执行请求req时所携带的本事务全局状态(包括TID、Lowts、Uppts、SI、Status=Running),反之,如果查询到了相应的全局状态信息,说明目标事务T已访问过本数据节点设备,则更新该数据节点设备上目标事务的全局状态信息,具体更新规则为:将目标事务T的逻辑执行生命周期的时间戳下界T.Lowts调整至大于或等于执行请求req中所携带的时间戳下界req.Lowts,将目标事务T的逻辑执行生命周期的时间戳上界T.Uppts调整至小于或等于执行请求req中所携带的时间戳上界req.Uppts,也即是说,令 $T.Lowts = \max(T.Lowts, req.Lowts)$ 、 $T.Uppts = \min(T.Uppts, req.Uppts)$ 。

[0218] 602、数据节点设备响应于对该逻辑执行生命周期校验通过,执行该目标事务。

[0219] 在上述过程中,数据节点设备在获取逻辑执行生命周期之后,对逻辑执行生命周期的合法性进行校验,若校验通过,执行该目标事务,否则,若校验不通过,则可以将全局状态信息中的事务状态Status置为正在回滚Aborting状态,并将修改后的全局状态信息发送至协调节点设备,以触发全局回滚该目标事务。

[0220] 在一些实施例中,在校验逻辑执行生命周期的合法性时,可以判断逻辑执行生命周期的时间戳下界是否小于逻辑执行生命周期的时间戳上界,如果时间戳下界小于或等于时间戳上界,确定校验通过,否则,如果时间戳下界大于时间戳上界,确定校验不通过。

[0221] 603、数据节点设备根据该目标事务的读集中的目标数据项以及写集中的待写入数据项,调整该逻辑执行生命周期,该目标数据项为符合该目标事务的查询条件且相对于该目标事务可见的数据项。

[0222] 在上述过程中,在事务执行阶段中,在接收到目标事务的执行请求之后,数据节点设备可以根据该目标事务的读集中的目标数据项,调整逻辑执行生命周期,而在事务验证阶段中,在接收到目标事务的验证请求之后,数据节点设备可以根据该目标事务的写集中的待写入数据项,调整逻辑执行生命周期,最终得到调整后的逻辑执行生命周期,此时需要通过下述步骤604对调整后的逻辑执行生命周期再次进行合法性校验,在校验通过时才准许提交目标事务,否则,一旦校验不通过仍然需要回滚目标事务。针对逻辑执行生命周期的调整策略将在下一个实施例中进行详述,这里不做展开说明。

[0223] 604、数据节点设备响应于对调整后的逻辑执行生命周期校验通过,提交该目标事务。

[0224] 在上述过程中,数据节点设备在验证完毕之后,将调整后的逻辑执行生命周期以及事务状态在全局状态信息中进行对应的更新,将更新后的全局状态信息、目标事务的读写集合以及操作是否成功的返回值等数据发送至协调节点设备,协调节点设备接收到各个数据节点设备的返回信息之后,进行对目标事务进行冲突验证,具体地,协调节点设备先检测各个数据节点设备返回的全局状态信息中的事务状态Status是否为正在回滚Aborting

状态,如果存在任一全局状态信息的Status=Aborting,进行全局回滚阶段,否则,更新目标事务的事务状态,并继续校验各个数据节点设备返回的全局状态信息中的调整后的逻辑执行生命周期,在对调整后的逻辑执行生命周期校验通过之后,协调节点设备向各个数据节点设备发送目标事务的提交指令,数据节点设备响应于该提交指令,提交该目标事务,否则,若对调整后的逻辑执行生命周期校验不通过,那么协调节点设备向各个数据节点设备发送目标事务的回滚指令,数据节点设备响应于该回滚指令,回滚该目标事务。

[0225] 在上述步骤604中对调整后的逻辑执行生命周期进行合法性校验的过程,与上述步骤602中对原本的逻辑执行生命周期进行合法性校验的过程类似,这里不做赘述。

[0226] 上述所有可选技术方案,可以采用任意结合形成本公开的可选实施例,在此不再一一赘述。

[0227] 本申请实施例提供的方法,通过响应于目标事务的执行请求,获取目标事务的逻辑执行生命周期,在对该逻辑执行生命周期校验通过之后,执行目标事务,在事务执行或者验证阶段中,根据目标事务的读集中的目标数据项以及写集中的待写入数据项,对逻辑执行生命周期进行调整,在对调整后的逻辑执行生命周期校验通过之后,提交目标事务,这种基于逻辑执行生命周期来处理事务的机制,无需依赖于锁机制,因此更加适用于读多写少的互联网应用场景,有利于提升分布式数据库系统的事务吞吐量,能够提升整个分布式数据库系统的事务处理性能。

[0228] 在上述实施例中,提供了基于逻辑执行生命周期来处理事务的机制,无需依赖于锁机制,能够避免锁机制对事务吞吐量的限制,而基于本申请实施例提供的多级可串行化隔离级别,可以针对不同的可串行化隔离级别提供不同的逻辑执行生命周期的调整策略,从而使得分布式系统具备同时支持多级可串行化隔离级别的能力,具体的调整策略以及事务执行流程将在本申请实施例中进行介绍。

[0229] 图7是本申请实施例提供的一种事务处理方法的交互流程图,参见图7,本申请实施例中针对上述实施例中事务执行阶段的步骤504中如何执行目标事务来进行展开说明,此外,还针对上述实施例中事务验证阶段的步骤507中如何验证目标事务来进行展开说明,该实施例包括:

[0230] 701、数据节点设备响应于目标事务的执行请求,获取该目标事务的全局状态信息,该全局状态信息用于表示该目标事务当前所处的执行状态。

[0231] 在一些实施例中,该全局状态信息可以表示为一个{TID,Lowts,Uppts,Lts,Gts,SI,Status}形式的七元组。TID为目标事务的事务标识,是一个全局唯一事务编号;Lowts为目标事务的逻辑执行生命周期的时间戳下界;Uppts为目标事务的逻辑执行生命周期的时间戳上界;Lts为目标事务的逻辑提交时间戳,该逻辑提交时间戳用于表示事务的逻辑提交时刻;Gts为目标事务的全局提交时间戳,该全局提交时间戳用于表示事务的全局提交时刻;SI为目标事务的事务快照时间戳,该事务快照时间戳用于表示事务的全局开始时刻;Status用于描述目标事务的事务状态,该事务状态Status可以如下包括7种类型:正在运行状态(Running)、正在验证状态(Validating)、验证通过状态(Validated)、正在提交状态(Committing)、提交完成状态(Committed)、正在回滚状态(Aborting)和回滚完成状态(Aborted)。

[0232] 上述执行请求中可以携带目标事务的状态参数,例如该状态参数包括上述TID、

Lowts、Uppts、SI或者Status=Running中至少一项,而由于Lts和Gts是在事务提交完成后才进行赋值的,因此Lts和Gts可以置为空或置为默认值,且在执行请求中无需携带Lts和Gts。

[0233] 在一些实施例中,数据节点设备在接收到协调节点设备发送的目标事务的执行请求(req)之后,根据目标事务的事务标识,查询在本地数据库中是否存储目标事务的全局状态信息,若查询不到目标事务的全局状态信息,可以初始化该目标事务的全局状态信息,将该初始化的全局状态信息赋值为执行请求(req)所携带的状态参数;反之,若查询到了目标事务的全局状态信息,说明当前的目标事务已访问过本数据节点设备,那么可以更新当前的数据节点设备上目标事务的事务全局状态信息,具体地更新方法可以为:将目标事务的逻辑执行生命周期的时间戳下界T.Lowts更新为查询到的时间戳下界T.Lowts与执行请求中所携带状态参数中的时间戳下界req.Lowts中的最大值,也即是说,令 $T.Lowts = \max(T.Lowts, req.Lowts)$,此外还将目标事务的逻辑执行生命周期的时间戳上界T.Uppts更新为查询到的时间戳上界T.Uppts与执行请求中所携带状态参数中的时间戳上界T.Uppts中的最小值,也即是说,令 $T.Uppts = \min(T.Uppts, req.Uppts)$ 。

[0234] 可选地,数据节点设备可以在缓存中开辟一段空间来存储各个活跃事务的全局状态信息,在接收到目标事务的执行请求(req)之后,解析执行请求,得到目标事务的状态参数,该状态参数包括TID、Lowts、Uppts、SI或者Status=Running中至少一项,数据节点设备可以以目标事务的事务标识TID为索引,在缓存中查询目标事务的全局状态信息,若该索引未能命中任一索引内容,说明查询不到目标事务的全局状态信息,此时将执行请求中的状态参数赋值为目标事务的全局状态信息,否则,若该索引能够命中任一索引内容,那么需要比较查询到的全局状态信息中的T.Lowts与执行请求所携带的状态参数中的req.Lowts,将两者中的最大值更新为最终的T.Lowts,此外,比较查询到的全局状态信息中的T.Uppts与执行请求所携带的状态参数中的req.Uppts,将两者中的最小值更新为最终的T.Uppts。

[0235] 702、数据节点设备基于该全局状态信息确定目标事务的逻辑执行生命周期,校验该逻辑执行生命周期,该逻辑执行生命周期用于表示所述目标事务在事务处理过程中的逻辑时间戳区间。

[0236] 在上述过程中,由于该全局状态信息包括该逻辑执行生命周期的时间戳下界T.Lowts和时间戳上界T.Uppts,那么通过检测T.Lowts是否小于或等于T.Uppts,从而可以完成对该逻辑执行生命周期的合法性校验,以判断出对该逻辑执行生命周期是否校验通过。

[0237] 在一些实施例中,响应于该逻辑执行生命周期的时间戳下界T.Lowts小于或等于该逻辑执行生命周期的时间戳上界T.Uppts,确定对该逻辑执行生命周期校验通过,执行下述步骤703;否则,响应于该逻辑执行生命周期的时间戳下界T.Lowts大于该逻辑执行生命周期的时间戳上界T.Uppts,确定对该逻辑执行生命周期校验不通过,此时可以将全局状态信息中的事务状态Status更新为正在回滚Aborting状态,也即令 $T.Status = Aborting$ 。

[0238] 703、数据节点设备响应于对该逻辑执行生命周期校验通过,执行该目标事务。

[0239] 在事务执行阶段中,数据节点设备需要对读写两种操作的执行计划进行处理,而按照目标事务所涉及的读写操作的类型不同,其执行流程也不尽相同,以下将分别针对读操作和写操作进行分类讨论。

[0240] 一、写操作的执行流程

[0241] 若该目标事务涉及针对数据项的写入操作,数据节点设备可以基于该执行请求,生成待写入数据项,将该待写入数据项存储到该目标事务的写集中。也即是说,数据节点设备根据目标事务的执行计划,生成需要插入/更新的数据项,将该数据项放入目标事务的写集结构中,写集结构已在上述实施例之前进行说明,这里不做赘述。

[0242] 可选地,这里的写集可以是本地写集,也可以是全局写集,在本申请实施例中以该写集为本地写集为例进行说明,能够避免因同步全局写集而带来的通信开销。

[0243] 在一些实施例中,由于在一些写写冲突率较高的业务负载下,分布式系统通常存在事务回滚率较高的问题,为降低事务回滚率,可以通过意向写技术来进行系统优化。具体地,用户可以通过全局变量定义一个意向写阈值,当某一数据项上的并发写事务数目超过意向写阈值时,分布式数据库系统对该数据项启用意向写技术。

[0244] 在意向写技术中,需要在数据项集合的页眉(Header)结构新增一个属性:写意向队列(IWlist),写意向队列用于表示当前正在等待更新本数据项的事务集合。需要说明的是,写意向队列IWlist与待写事务WT之间的区别在于,写意向队列IWlist是一个列表,在列表中可以记录一个或多个事务标识TID,而待写事务WT中通常会记载单个事务标识TID。

[0245] 在读取阶段中,如果意向写队列技术被启用,当多个并发事务试图修改同一个数据项时,只有一个事务被允许修改该数据项,并对该数据项加排它锁,其他事务的事务标识TID将被添加到意向写队列中(该队列可以为一个先进先出队列),然后进入等待状态。在上述事务提交/回滚完成后,该数据项上的排它锁释放,并唤醒位于意向写队列队尾的事务标识TID所对应的事务。当数据项上的并发写事务数目降至意向写阈值以下时,意向写技术对该数据项失效,当意向写队列中的所有事务都执行完成后,队列空间被释放。

[0246] 需要说明的是,意向写技术可能会出现死锁问题,假设数据项x和y分别位于不同数据节点设备RM1和RM2上,事务T1和T2并发同时更新数据项x和y,那么在RM1和RM2上基于意向写技术的操作如下:

[0247] 在RM1上,事务T1首先申请到数据项x的排它锁;随后,事务T2申请更新数据项x,因检测到有事务T1正在更新数据项x,所以事务T2被加入到数据项x的意向写队列中,等待事务T1提交后被唤起继续执行。

[0248] 在RM2上,事务T2首先申请到数据项y的排它锁;随后,事务T1申请更新数据项y,因检测到有事务T2正在更新数据项y,所以事务T1被加入到数据项y的意向写队列中,等待事务T2提交后被唤起继续执行。

[0249] 此时会出现事务T1和事务T2相互等待的问题,即产生了死锁。为避免死锁问题而导致分布式系统性能下降,可以设置一个超时等待机制,也即是说,如果事务T在意向写队列中的等待时长超过了系统锁的超时时长,则事务T会选择回滚自身,其中,该系统锁的超时时长由技术人员进行设定,可以是任一大于或等于0的数值,本申请实施例不对超时时长进行具体限定。

[0250] 二、读操作的执行流程

[0251] 若该目标事务涉及针对数据项的读取操作,数据节点设备可以基于该执行请求中的查询条件,确定该查询条件所对应的至少一个待判断数据项;从该至少一个待判断数据项中,确定相对于该目标事务可见的目标数据项,将该目标数据项存储到该目标事务的读

集中。

[0252] 可选地,这里的读集可以是本地读集,也可以是全局读集,在本申请实施例中以该读集为本地读集为例进行说明,能够避免因同步全局读集而带来的通信开销。

[0253] 在上述过程中,由于分布式系统中涉及到多级可串行化隔离级别,那么对于涉及读操作的目标事务,其执行流程为:根据给定的查询条件(也即读取条件),定位到待查询数据项(也即待判断数据项),根据当前设置的可串行化隔离级别,执行对应级别的可见性判断算法,判断出待查询数据项中的可见数据(也即目标数据项)。

[0254] 下面,将分别针对不同可串行化隔离级别下的可见性判断算法进行介绍,为了描述方便,将目标事务表示为T。

[0255] 1、顺序并发结果可串行化SCR隔离级别的可见性判断算法

[0256] 数据节点设备响应于数据库系统处于顺序并发结果可串行化SCR隔离级别,对该至少一个待判断数据项中任一待判断数据项,若产生该待判断数据项的事务的逻辑提交时间戳 $v.Lts$ 小于该目标事务的事务快照时间戳 $T.SI$,确定该待判断数据项为候选数据项;将具有相同主键标识的候选数据项中逻辑提交时间戳最大的候选数据项确定为目标数据项。其中,在该顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定该两个并发事务的先后关系,该逻辑提交时间戳用于表示事务的逻辑提交时刻,该事务快照时间戳用于表示事务的全局开始时刻。

[0257] 在一些实施例中,从数据版本的角度出发,数据节点设备可以根据用户给定的查询条件,定位到待判断可见性的数据项(也即至少一个待判断数据项),由于属于同一个数据项集合的多个数据版本可以按照时间戳从新到旧的顺序进行存放,那么可以从最新版本开始遍历查找,对于任一数据版本 v ,如果数据库系统处于SCR隔离级别,数据节点设备可以判断数据版本 v 的逻辑提交时间戳 $v.Lts$ 是否小于目标事务的事务快照时间戳 $T.SI$ (也即目标事务开始时获取的全局时间戳),换言之,判断 $v.Lts < T.SI$ 是否成立,若成立,确定该数据版本 v 可见,退出遍历循环,否则,跳转至下一个较老的数据版本,重复执行判断步骤。

[0258] 2、顺序并发结果可串行化SCR隔离级别之外的可串行化隔离级别的可见性判断算法

[0259] 数据节点设备响应于数据库系统处于除了顺序并发结果可串行化SCR隔离级别之外的可串行化隔离级别(包括线性可串行化LS隔离级别、线性并发结果可串行化LCR隔离级别、线性点可串行化LP隔离级别或者顺序可串行化SR隔离级别中任一项),对该至少一个待判断数据项中任一待判断数据项,若产生该待判断数据项的事务的全局提交时间戳 $v.Gts$ 小于该目标事务的事务快照时间戳 $T.SI$,确定该待判断数据项为候选数据项;将具有相同主键标识的候选数据项中全局提交时间戳最大的候选数据项确定为目标数据项。其中,在该顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定该两个并发事务的先后关系,该全局提交时间戳用于表示事务的全局提交时刻,该事务快照时间戳用于表示事务的全局开始时刻。

[0260] 在一些实施例中,从数据版本的角度出发,数据节点设备可以根据用户给定的查询条件,定位到待判断可见性的数据项(也即至少一个待判断数据项),由于属于同一个数据项集合的多个数据版本可以按照时间戳从新到旧的顺序进行存放,那么可以从最新版本开始遍历查找,对于任一数据版本 v ,如果数据库系统处于LS、LCR、LP或者SR中任一隔离级

别,数据节点设备可以判断数据版本 v 的全局提交时间戳 $v.Gts$ 是否小于目标事务的事务快照时间戳 $T.SI$ (也即目标事务开始时获取的全局时间戳),换言之,判断 $v.Gts < T.SI$ 是否成立,若成立,确定该数据版本 v 可见,退出遍历循环,否则,跳转至下一个较老的数据版本,重复执行判断步骤。

[0261] 704、在目标事务的执行过程中,根据该目标事务的读集中的目标数据项,调整该逻辑执行生命周期,该目标数据项为符合该目标事务的查询条件且相对于该目标事务可见的数据项。

[0262] 在上述步骤704中,在确定出相对于目标事务可见的目标数据项之后,将目标数据项存储到目标事务的读集中,对于读集中的任一目标数据项,数据节点设备可以执行下述子步骤7041-7044,以根据目标数据项调整该逻辑执行生命周期。

[0263] 7041、数据节点设备将目标事务的事务标识 TID 写入到读取的目标数据项对应的活跃事务集合 $RTlist$ 中。

[0264] 其中, $RTlist$ 是目标数据项所属数据项集合的页眉(Header)结构中一个属性值, $RTlist$ 中记录了访问过该数据项集合中最新数据版本的活跃事务集合,该活跃事务集合可以是数组的形式,也可以是列表、队列、堆栈等形式,本申请实施例不对活跃事务集合的形式进行具体限定, $RTlist$ 中每个元素可以是读取过上述最新数据版本的事务的事务标识(TID)。

[0265] 7042、数据节点设备将该逻辑执行生命周期的时间戳下界 $T.lowst$ 调整至大于产生该目标数据项的事务的逻辑提交时间戳 $v.Lts$,该逻辑提交时间戳用于表示事务的逻辑提交时刻。

[0266] 在上述过程中,数据节点设备调整目标事务的逻辑执行生命周期的时间戳下界 $T.Lowts$,使得调整后的时间戳下界 $T.Lowts$ 大于所读取到的目标数据项上记录的逻辑提交时间戳 $v.Lts$,其中,由于在上述实施例之前介绍了本申请实施例中数据项的基本数据结构,在每个数据项 v 的键中均记录了 $\langle User_key, Lts, Gts \rangle$,因此只需要调整 $T.Lowts$ 大于目标数据项 v 的键中记载的 Lts 即可。

[0267] 在一些实施例中,数据节点设备针对时间戳下界 $T.Lowts$ 的调整方式可以包括下述至少一项:

[0268] 1、数据节点设备响应于任一目标数据项的逻辑提交时间戳 $v.Lts$ 小于该时间戳下界 $T.Lowts$,将该时间戳下界 $T.Lowts$ 确定为调整后的时间戳下界。也即是说,若 $T.Lowts > v.Lts$,将 $T.Lowts$ 置为 $T.Lowts$ 不变。

[0269] 2、数据节点设备响应于任一目标数据项的逻辑提交时间戳 $v.Lts$ 等于该时间戳下界 $T.Lowts$,将该目标数据项的最终提交时间戳 $v.cts$ 加一所得的数值确定为调整后的时间戳下界。也即是说,若 $T.Lowts = v.Lts$,将 $T.Lowts$ 置为 $v.cts+1$ (令 $T.Lowts = v.cts+1$)。其中,该最终提交时间戳 $v.cts$ 处于产生目标数据项的事务的逻辑执行生命周期 $[Lowts, Uppts)$ 中。

[0270] 3、数据节点设备响应于任一目标数据项的逻辑提交时间戳 $v.Lts$ 大于该时间戳下界 $T.Lowts$,将该目标数据项的最终提交时间戳 $v.cts$ 加一所得的数值确定为调整后的时间戳下界。也即是说,若 $T.Lowts < v.Lts$,将 $T.Lowts$ 置为 $v.cts+1$ (令 $T.Lowts = v.cts+1$)。其中,该最终提交时间戳 $v.cts$ 处于产生目标数据项的事务的逻辑执行生命周期 $[Lowts,$

Uppts) 中。

[0271] 上述三种方式可以统一表示为下述公式：

$$[0272] \quad T.Lowts = \begin{cases} T.Lowts & (T.Lowts > v.Lts) \\ v.cts + 1 & (T.Lowts = v.Lts) \\ v.cts + 1 & (T.Lowts < v.Lts) \end{cases}$$

[0273] 7043、数据节点设备将该逻辑执行生命周期的时间戳上界T.Uppts调整至小于或等于产生第一数据项的事务的逻辑提交时间戳v_new.Lts,该第一数据项为与该目标数据项具有相同主键标识的下一数据项。

[0274] 在上述过程中,数据节点设备调整目标事务的逻辑执行生命周期的时间戳上界T.Uppts,使得调整后的时间戳上界T.Uppts小于或等于所读取到的目标数据项的下一个新版本上记录的逻辑提交时间戳v_new.Lts,其中,由于在上述实施例之前介绍了本申请实施例中数据项的基本数据结构,在每个数据项v的键中均记录了<User_key,Lts,Gts>,因此只需要根据数据项v确定出与该数据项v具有相同主键标识的下一个新版本v_new(第一数据项),进而调整T.Uppts小于或等于新版本v_new的键中记载的逻辑提交时间戳v_new.Lts即可。

[0275] 具体地,数据节点设备可以将该时间戳上界T.Uppts与产生该第一数据项v_new的事务的逻辑提交时间戳v_new.Lts之间的最小值确定为调整后的时间戳上界,也即是说,令T.Uppts=min(T.Uppts,v_new.Lts)。

[0276] 7044、数据节点设备响应于该目标数据项对应的待写事务WT不为空,将该逻辑执行生命周期的时间戳上界T.Uppts调整至小于或等于该待写事务WT的时间戳下界WT.Lowst。

[0277] 在上述过程中,数据节点设备可以判断当前数据版本(目标数据项)对应的数据项页眉结构中的WT字段是否为0,WT表示该数据项所对应的待写事务,WT中可以记录当前要写入该数据项的事务的事务标识(TID)。若WT字段不为0,那么需要将原本的时间戳上界T.Uppts与WT内记载事务的时间戳下界WT.Lowst之间的最小值确定为调整后的时间戳上界,也即是说,令T.Uppts=min(T.Uppts,WT.Lowts)。

[0278] 需要说明的是,响应于数据库系统处于线性可串行化LS、线性并发结果可串行化LCR或者顺序可串行化SR中任一隔离级别,若该目标事务的写集中包括与该目标数据项具有相同主键标识的待写入数据项,回滚该目标事务。

[0279] 也即是说,在LS、LCR或者SR中任一隔离级别下,在读到某个目标数据项之后,还需要额外进行如下操作:如果写集中存在该目标数据项的下一个新版本,那么更新该目标事务的本地事务状态为正在回滚Aborting状态,即更新T.Status=Aborting,并回滚该目标事务,用来保证目标事务T的全局提交时间戳T.Gts和逻辑提交时间戳T.Lts所确定的顺序一致。而对于只读事务来说,则无需执行上述额外的操作,这样能够减少只读事务的回滚情况。

[0280] 705、数据节点设备向协调节点设备返回目标事务的执行结果。

[0281] 在上述过程中,数据节点设备向协调节点设备发送执行结果(res,可视为一种返回消息),同时还可以在执行结果中封装当前数据节点设备上目标事务的全局状态信息、当

前读写操作的结果集(包括本地读集和本地写集)以及操作是否成功的返回值等,本申请实施例不对执行结果中包含的内容进行具体限定。

[0282] 上述步骤705与上述步骤504类似,这里不做赘述。

[0283] 706、协调节点设备向数据节点设备发送目标事务的验证请求。

[0284] 上述步骤706与上述步骤506类似,这里不做赘述。

[0285] 在上述过程中,协调节点设备接收到各个数据节点设备的执行结果之后,首先检查各个执行结果所携带的全局状态信息中事务状态是否为正在回滚Aborting状态,如果是,那么进入全局回滚阶段,否则,继续校验执行结果所携带的逻辑执行生命周期是否合法,如果校验通过(检测到 $T.Lowts \leq T.Uppts$),代表本次操作执行完毕,进入事务验证阶段,协调节点设备将目标事务的事务状态Status更新为正在验证Validating状态,执行上述步骤706中发送验证请求的操作,其中,该验证请求中封装了Status=Validating,否则,如果校验不通过(检测到 $T.Lowts > T.Uppts$),那么仍然会进入到全局回滚阶段。上述针对逻辑执行生命周期进行合法性校验的过程与上述步骤702类似,这里不做赘述。

[0286] 从上述事务执行阶段可以看出,在执行目标事务的过程中,通信主要在目标事务的协调节点设备和相关数据节点设备之间发生,目标事务每成功读取一次数据需要两次通信:目标事务的协调节点设备发送请求信息到相关的数据节点设备上、相关的数据节点设备返回结果给协调节点设备。因此,在事务执行阶段,假设n为远程读取的次数,那么最多需要进行2n次通信,最大通信量可以表示为 $n \times (\text{请求消息大小} + \text{响应消息大小})$ 。

[0287] 707、数据节点设备响应于该验证请求,对目标事务进行冲突验证,在冲突验证过程中,根据该目标事务的写集中的待写入数据项,调整该逻辑执行生命周期。

[0288] 在传统的OCC(Optimistic Concurrency Control,乐观并发控制)冲突验证算法中,是将待验证事务的读集和已经完成事务的写集进行比较,这样会导致验证阶段产生大量的事务回滚。在本申请实施例中,采用了动态调整事务可串行化顺序的思想,来对事务验证阶段进行优化,由于局部事务的验证算法可以基于全局事务的验证算法进行少量调整获得,因此在本申请实施例中,以全局事务为例对具体地验证算法进行介绍:

[0289] 7071、数据节点设备响应于目标事务的验证请求,更新该目标事务的全局状态信息中的事务状态Status。

[0290] 也即是说,数据节点设备解析验证请求,将验证请求中携带的事务状态Status的取值赋值给本地存储的事务状态Status。由于验证请求中封装了Status=Validating,因此数据节点设备实际上会将本地存储的事务状态更新为Validating。

[0291] 7072、数据节点设备在冲突验证过程中,根据该目标事务的写集中的待写入数据项,调整该逻辑执行生命周期,得到调整后的逻辑执行生命周期。

[0292] 其中,调整后的逻辑执行生命周期与该写集中待写入数据项的读事务不存在读写冲突。

[0293] 在一些实施例中,在事务验证阶段对目标事务的逻辑执行生命周期进行调整,是为了防止由于读写冲突而造成的回滚,遍历本地写集中的每个元素(也即每个待写入数据项),对目标事务的逻辑执行生命周期进行如下调整:

[0294] A) 数据节点设备获取该写集中各个待写入数据项的最大读事务时间戳Rts。

[0295] 上述写集可以是本地写集,也可以是全局写集,在本申请实施例中以该写集为本

地写集为例进行说明,这样能够避免因同步全局写集而带来的通信开销。

[0296] 其中,对任一待写入数据项,该待写入数据项的最大读事务时间戳Rts用于表示读取过该待写入数据项的各事务的逻辑提交时间戳中的最大值,该最大读事务时间戳Rts记录在每个待写入数据项所对应的页眉结构中。

[0297] B) 数据节点设备确定各个待写入数据项的最大读事务时间戳Rts中的最大值。

[0298] C) 数据节点设备将该逻辑执行生命周期的时间戳下界调整至大于或等于各个待写入数据项的最大读事务时间戳Rts中的最大值。

[0299] 在上述过程中,数据节点设备可以响应于该逻辑执行生命周期的时间戳下界T.Lowst大于该最大值v.rts,将该逻辑执行生命周期的时间戳下界确定为调整后的时间戳下界;响应于该逻辑执行生命周期的时间戳下界T.Lowst等于该最大值v.rts,将该最大值加一所得的数值v.rts+1确定为调整后的时间戳下界;响应于该逻辑执行生命周期的时间戳下界T.Lowst小于该最大值v.rts,将该最大值v.rts确定为调整后的时间戳下界。

[0300] 上述步骤A) -C) 中,逻辑执行生命周期的调整规则可以表示如下公式:

$$[0301] \quad T.Lowts = \begin{cases} T.Lowts & (T.Lowts > v.rts) \\ v.rts + 1 & (T.Lowts = v.rts) \\ v.rts & (T.Lowts < v.rts) \end{cases}$$

[0302] 7073、数据节点设备在待写入数据项的待写事务WT中写入目标事务的事务标识TID。

[0303] 在上述过程中,数据节点设备对各个待写入数据项的待写事务WT进行修改,在待写事务WT中写入当前处理的目标事务的事务标识TID,在写入时可以使用无锁的CAS (Compare And Swap,比较与交换,一种无锁算法) 技术为WT进行赋值,以提高分布式数据库系统的性能;如果待写入数据项的待写事务WT不为0,则将目标事务T的事务状态Status置为正在回滚状态Aborting,并直接向协调节点设备返回验证失败信息。

[0304] 7074、对该写集中任一待写入数据项的读事务列表中的任一读事务,数据节点设备响应于该读事务处于验证通过状态或提交完成状态,将该逻辑执行生命周期的时间戳下界调整至大于或等于该读事务的时间戳上界。

[0305] 在上述过程中,数据节点设备可以获取该写集中待写入的数据项所对应活跃事务列表中读事务T_c的事务状态。其中,活跃事务列表中读事务T_c的数量可以是一个或多个,本申请实施例不对读事务T_c的数量进行具体限定。

[0306] 对于每个待写入数据项的活跃事务列表中的每个读事务T_c,如果读事务T_c处于验证通过Validated状态或提交完成Committed状态,数据节点设备执行上述步骤7074,将读事务的时间戳上界T_c.Uppts以及原本时间戳下界T.Lowts中的最大值确定为调整后的时间戳下界,也即令T.Lowts=max(T_c.Uppts,T.Lowts),从而实现将时间戳下界调整至大于或等于读事务T_c的时间戳上界;对于事务状态为正在运行Running状态的读事务T_c,则执行下述步骤7075,这里不做赘述。

[0307] 7075、数据节点设备响应于该读事务处于正在运行状态,在不同的可串行化隔离级别下,对该逻辑执行生命周期进行不同的调整。

[0308] 在上述过程中,针对不同的可串行化隔离级别,数据节点设备可以具有如下几种

调整策略：

[0309] A) 响应于数据库系统处于线性可串行化LS、线性并发结果可串行化LCR或者顺序可串行化SR中任一隔离级别，数据节点设备在等待该读事务 T_c 结束后继续处理该目标事务T，将该逻辑执行生命周期的时间戳下界 $T.Lowts$ 调整至大于或等于该读事务的时间戳上界 $T_c.Uppts$ 。

[0310] 也即是说，在LS、LCR或者SR中任一隔离级别下，需要等待读事务 T_c 结束后，本目标事务T再继续执行，并调整： $T.Lowts = \max(T_c.Uppts, T.Lowts)$ 。

[0311] B) 响应于数据库系统处于线性点可串行化LP隔离级别，数据节点设备回滚该目标事务。

[0312] 也即是说，在LP隔离级别下，数据节点设备需要将目标事务T的事务状态Status置为正在回滚Aborting状态，并将本目标事务T全局回滚。

[0313] C) 响应于数据库系统处于顺序并发结果可串行化SCR隔离级别，数据节点设备将该逻辑执行生命周期的时间戳下界 $T.Lowts$ 调整至大于或等于该读事务的时间戳下界 $T_c.Lowts$ ，将该读事务的时间戳上界 $T_c.Uppts$ 调整至小于或等于该逻辑执行生命周期的时间戳下界 $T.Lowts$ 。

[0314] 也即是说，在SCR隔离级别下，数据节点设备进行如下调整：如果该读事务的时间戳下界 $T_c.Lowts$ 等于该逻辑执行生命周期的时间戳下界 $T.Lowts$ ，将该读事务的时间戳下界加一所得的数值 $T_c.Lowts+1$ 确定为调整后的时间戳下界；如果该读事务的时间戳下界 $T_c.Lowts$ 大于该逻辑执行生命周期的时间戳下界 $T.Lowts$ ，将该读事务的时间戳下界 $T_c.Lowts$ 确定为调整后的时间戳下界；然后，将该读事务的时间戳上界调整为该读事务的时间戳上界 $T_c.Uppts$ 与调整后的时间戳下界 $T.Lowts$ 中的最小值。

[0315] 也即是说，对于每个待写入数据项的活跃事务列表中的每个读事务 T_c ，如果读事务 T_c 处于正在运行状态Running，响应于 $T_c.Lowts$ 等于 $T.Lowts$ ，那么调整 $T.Lowts = T_c.Lowts+1$ ；响应于 $T_c.Lowts$ 大于 $T.Lowts$ ，那么调整 $T.Lowts = T_c.Lowts$ 。进一步地，将调整读事务 T_c 的时间戳上界小于或等于目标事务T的时间戳下界，也即是说，令 $T_c.Uppts = \min(T_c.Uppts, T.Lowts)$ 。

[0316] 7076、数据节点设备对目标事务调整后的逻辑执行生命周期进行校验。

[0317] 在上述过程中，在事务验证阶段为避免读写冲突，根据写集中待写入数据项，对目标事务的逻辑执行生命周期进行了修改，得到了调整后的逻辑执行生命周期，此时仍然需要对调整后的逻辑执行生命周期进行再次的合法性校验，也即是说，检测调整后的逻辑执行生命周期的时间戳下界是否仍小于或等于调整后的逻辑执行生命周期的时间戳上界，响应于该时间戳下界小于或等于该时间戳上界($T.Lowts \leq T.Uppts$)，确定对该调整后的逻辑执行生命周期校验通过，将全局状态信息中的事务状态Status更新为验证通过Validated状态，也即令 $T.Status = Validated$ ；否则，响应于该时间戳下界大于该时间戳上界($T.Lowts > T.Uppts$)，确定对该调整后的逻辑执行生命周期校验不通过，此时可以将全局状态信息中的事务状态Status更新为正在回滚Aborting状态，也即令 $T.Status = Aborting$ ，并向协调节点设备返回验证失败信息。

[0318] 708、数据节点设备向协调节点设备返回目标事务的验证结果。

[0319] 在上述过程中，数据节点设备向协调节点设备返回本地的验证结果(res)，同时还

可以在验证结果中封装本地的目标事务的全局状态信息,该全局状态信息中包括调整后的逻辑执行生命周期。

[0320] 上述步骤708与上述步骤507类似,这里不做赘述。

[0321] 从上述事务验证阶段可以看出,在验证目标事务的过程中,通信主要在目标事务的协调节点设备和相关数据节点设备之间发生,通信包括两类:目标事务的协调节点设备向每个相关的数据节点设备发送验证请求及本地写集、相关的数据节点设备反馈本地的验证结果给协调节点设备。因此,在事务验证阶段,假设 m 为与目标事务 T 相关的数据节点设备的数量,那么最多需要进行 $2m$ 次通信,最大通信量可以表示为 $m \times (\text{验证请求消息大小} + \text{验证结果消息大小}) + \text{全局写集大小}$ 。

[0322] 表1为多级可串行化隔离级别的实现机制总结表,请参考表1,将不同可串行化隔离级别下实现机制进行了总结,可以看出,随着可串行化隔离级别的降低,一些系统开销可以被省去(相当于一些条件被放宽了),因此事务处理性能也就会随之提高。

[0323] 表1

[0324]

	线性可串行化 LS	线性并发可串行化 LCR	线性点可串行化 LP	顺序可串行化 SR	顺序并发结果可串行化 SCR
基本特征 (条件)	任何 2 个事务, 需要保证: $T1.Uppts < T2.Lowts$	任何 2 个并发事务, 需要保证: $T1.Uppts < T2.Lowts$	同左	并发或并行事务, 如果没有冲突允许 $T2.Uppts > T1.Uppts > T2.Lowts$	同左
排序	并发控制排序顺序决定从全局获取时间戳的顺序, 先确定了可提交, 再获取全局时间戳作为 Gts。需要保证: $T1.Gts < T2.Gts$, 且 $T1.Lts < T2.Lts$	同左	同左	并发控制排序顺序决定从全局获取时间戳的顺序。 需要保证: $T1.Gts < T2.Gts$; 且如果 $T1$ 和 $T2$ 是并发事务, 则 $T1.Lts < T2.Lts$	$T1.Lts < T2.Lts$ 不需要获取 Gts
读数据可见性判断	以上面 2 种排序都满足为可见	以 Commit-time(提交时间)排序满足为可见	同左	以 Commit-time(提交时间)排序满足为可见	以 DTA 排序满足为可见
SI	全局(从全局获取, 有通讯代价, 后同)	同左	同左	LC(全局 LC), HLC(全局或本地 HLC)	同左

[0325]	Gts				
	全局 提交 时间 戳	全局	全局	全局, 但等于 SI	全局 不需要获取

[0326] 在上表1中,基本特征指的是事务在验证是否可以提交时,为了保证可串行化隔离级别之外,还需要额外保证的条件;排序即指的是在事务验证阶段中,为了保证所规定的基本特征而制定的排序规则,排序是事务的提交序。

[0327] 可以看出,在SR或SCR中任一隔离级别下,可以采用HLC机制来部分替代全局时间戳生成集群的功能,从而消除与全局时间戳生成集群之间的一轮通信交互,以提升事务执行效率。

[0328] 709、协调节点设备汇总数据节点设备的验证结果,确定目标事务的全局验证结果。

[0329] 协调节点设备在接收到所有相关数据节点设备反馈的本地验证结果之后,需要判断目标事务T是进入提交阶段还是回滚阶段,判断方法可以如下:

[0330] 如果在所有的验证结果中,不存在事务状态Status被置为正在回滚Aborting状态的验证结果,那么将所有相关数据节点设备上的调整后的逻辑执行生命周期(携带在验证结果中)求交集,得到新的时间戳区间 $[T.Lowts, T.Uppts)$,协调节点设备对上述新的时间戳区间进行合法性校验,如果对上述新的时间戳区间校验通过,确定全局验证结果为验证通过,选取当前时间戳区间的时间戳下界 $T.Lowts$ 作为目标事务T的逻辑提交时间戳 $T.Lts$,将目标事务的全局事务状态记为提交完成Committed状态,并向所有相关数据节点设备发送目标事务的提交指令;否则,如果对上述新的时间戳区间校验不通过,或者如果存在事务状态Status被置为正在回滚Aborting状态的验证结果,表明目标事务T没有通过验证,确定全局验证结果为验证不通过,则需要回滚目标事务T,此时协调节点设备将目标事务T的事务状态Status设置为回滚完成Aborted状态,并向所有相关数据节点设备发送目标事务的回滚指令。

[0331] 在一些实施例中,在不同的可串行化隔离级别下,目标事务T的全局提交时间戳 $T.Gts$ 的获取操作也不尽相同:对于LS、LCR或者SR中任一隔离级别,所有事务在提交时都需要与全局时间戳生成集群进行通信,获取当前的全局时间戳作为 $T.Gts$;在LP隔离级别下,事务的全局提交时间戳设置为 $T.SI$;在SCR隔离级别下,不需要获取全局提交时间戳,只需要得到逻辑提交时间戳,即 $T.Gts$ 为空。

[0332] 710、协调节点设备响应于全局验证结果为验证通过,向数据节点设备发送目标事务的提交指令。

[0333] 上述步骤710与上述步骤509类似,这里不做赘述。

[0334] 711、数据节点设备响应于对调整后的逻辑执行生命周期校验通过,提交该目标事务。

[0335] 在上述过程中,数据节点设备接收到协调节点设备的提交指令之后,可以执行下述几项操作中的至少一项:

[0336] 1) 数据节点设备将目标事务的写集中的数据落盘,并在每个新写入的数据项的key中拼入协调节点设备传来的逻辑提交时间戳Lts和全局提交时间戳Gts。

[0337] 2) 数据节点设备清理目标事务的事务执行上下文信息。

[0338] 具体地,数据节点设备可以将每个读集中目标数据项对应的最大读事务时间戳Rts修改为Rts与逻辑提交时间戳Lts两者中的最大值,从该目标数据项的活跃事务列表RTlist中将目标事务的事务标识TID删除。

[0339] 进一步地,数据节点设备还可以将每个写集中待写入数据项原本的Lts修改为目标事务的逻辑提交时间戳。

[0340] 进一步地,数据节点设备还可以将写集中待写入数据项对应的WT(待写事务)字段重置为0。

[0341] 进一步地,数据节点设备还可以清空目标事务T的读集和写集。

[0342] 在一些实施例中,由于有可能协调节点设备对目标事务验证不通过,此时发送的是回滚指令,在数据节点设备接收到回滚指令之后,仍然需要对事务执行上下文信息进行清理:

[0343] 具体地,数据节点设备可以在每个读集中目标数据项的活跃事务列表RTlist中将目标事务的事务标识TID删除。进一步地,还可以将写集中待写入数据项对应的WT(待写事务)字段重置为0。进一步地,还可以清空目标事务T的读集和写集。

[0344] 从上述情况可以看出,在目标事务T的提交/回滚阶段,通信主要在目标事务T的协调节点设备和相关数据节点设备之间发生,通信主要包含以下两类:目标事务T的协调节点设备向每个相关数据节点设备发送提交/回滚指令、每个相关数据节点设备向协调节点设备发送相应的提交/回滚完成消息。因此,提交/回滚阶段最多进行 $2m$ 次通信,通信量的大小为 $m \times (\text{提交/回滚指令消息大小} + \text{提交/回滚完成消息大小})$,其中 m 为目标事务T相关数据节点设备的个数。

[0345] 需要说明的是,在一些实施例中,本申请实施例提供的多级可串行化隔离级别适用于基于MVCC(Multi-Version Concurrency Control,多版本并发控制)机制的分布式数据库系统。应用本申请实施例提供的实施方案,系统可以获得如下两点收益:(1)提升系统事务处理的正确性。基于MVCC机制的分布式数据库系统,一般不提供可串行化级别的事务处理能力,使用本申请实施例提出的事务处理机制,系统可以较好地提供可串行化事务处理能力,从而保证系统事务处理的正确性。具体地,在提供了线性可串行化LS、线性并发结果可串行化LCR、线性点可串行化LP、顺序可串行化SR、顺序并发结果可串行化SCR这五种多级可串行化隔离级别的基础上,在数据结构上,为每个数据项构建 $\langle \text{User_key}, \text{Lts}, \text{Gts} \rangle$ 形式的Key,并将其余属性值作为Value,在读写集合的维护策略上,以Header结构的方式维护用户主键User_key、最大读事务时间戳Rts、待写事务WT以及读事务列表RTlist,对于每个待执行的事务,维护一个 $\{\text{TID}, \text{Lowts}, \text{Uppts}, \text{Lts}, \text{Gts}, \text{SI}, \text{Status}\}$ 形式的七元组作为该事务的全局状态信息。实施过程与前述过程相同,这里不做赘述。(2)提升系统事务处理的性能。本申请实施例提出的事务处理机制充分考虑了基于MVCC机制的分布式数据库系统提供的特性,从而通过引入OCC机制的事务处理机制以及一系列的优化方案,能够保证获得较好的系统性能。

[0346] 上述所有可选技术方案,可以采用任意结合形成本公开的可选实施例,在此不再

一一赘述。

[0347] 本申请实施例提供的方法,通过响应于目标事务的执行请求,获取目标事务的逻辑执行生命周期,在对该逻辑执行生命周期校验通过之后,执行目标事务,在事务执行或者验证阶段中,根据目标事务的读集中的目标数据项以及写集中的待写入数据项,对逻辑执行生命周期进行调整,在对调整后的逻辑执行生命周期校验通过之后,提交目标事务,这种基于逻辑执行生命周期来处理事务的机制,无需依赖于锁机制,因此更加适用于读多写少的互联网应用场景,有利于提升分布式数据库系统的事务吞吐量,能够提升整个分布式数据库系统的事务处理性能。

[0348] 进一步地,提出了分布式事务的多级可串行化隔离级别,为分布式事务处理的正确性提供了衡量标准,通过采用不同级别的可串行化,系统可以在可串行这个正确性前提下,提供多种不同的可串行化实现机制,从而提供可调节的事务处理性能。在此基础上,提出了一套分布式事务处理方案,使得系统具备同时支持多级可串行化的能力。

[0349] 进一步地,提出了一系列方法优化分布式事务的执行效率,减少事务处理时的额外执行开销和事务回滚开销,从而提升系统整体效率。比如,提出了基于范围的读写集维护策略,减少读写集维护开销,从而减少事务上下文的维护开销,又比如,提出了基于HLC的时间戳优化策略,以减少与全局时间戳生成集群的一轮通信开销,从而优化整个分布式系统的性能,又比如,提出了数据项时间戳缓存策略,减少数据项上的时间戳维护开销,最后,还提出了只读事务优化策略,保证只读事务不被回滚,降低系统内的事务回滚开销。

[0350] 图8是本申请实施例提供的一种事务处理装置的结构示意图,请参考图8,该装置包括:

[0351] 获取模块801,用于响应于目标事务的执行请求,获取该目标事务的逻辑执行生命周期,该逻辑执行生命周期用于表示该目标事务在事务处理过程中的逻辑时间戳区间;

[0352] 执行模块802,用于响应于对该逻辑执行生命周期校验通过,执行该目标事务;

[0353] 调整模块803,用于根据该目标事务的读集中的目标数据项以及写集中的待写入数据项,调整该逻辑执行生命周期,该目标数据项为符合该目标事务的查询条件且相对于该目标事务可见的数据项;

[0354] 提交模块804,用于响应于对调整后的逻辑执行生命周期校验通过,提交该目标事务。

[0355] 本申请实施例提供的装置,通过响应于目标事务的执行请求,获取目标事务的逻辑执行生命周期,在对该逻辑执行生命周期校验通过之后,执行目标事务,在事务执行或者验证阶段中,根据目标事务的读集中的目标数据项以及写集中的待写入数据项,对逻辑执行生命周期进行调整,在对调整后的逻辑执行生命周期校验通过之后,提交目标事务,这种基于逻辑执行生命周期来处理事务的机制,无需依赖于锁机制,因此更加适用于读多写少的互联网应用场景,有利于提升分布式数据库系统的事务吞吐量,能够提升整个分布式数据库系统的事务处理性能。

[0356] 在一种可能实施方式中,若该目标事务涉及针对数据项的读取操作,基于图8的装置组成,该执行模块802包括:

[0357] 确定单元,用于基于该执行请求中的查询条件,确定该查询条件所对应的至少一个待判断数据项;

[0358] 确定存储单元,用于从该至少一个待判断数据项中,确定相对于该目标事务可见的目标数据项,将该目标数据项存储到该目标事务的读集中。

[0359] 在一种可能实施方式中,该确定存储单元用于:

[0360] 响应于数据库系统处于顺序并发结果可串行化隔离级别,对该至少一个待判断数据项中任一待判断数据项,若产生该待判断数据项的事务的逻辑提交时间戳小于该目标事务的事务快照时间戳,确定该待判断数据项为候选数据项;将具有相同主键标识的候选数据项中逻辑提交时间戳最大的候选数据项确定为目标数据项;

[0361] 其中,在该顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定该两个并发事务的先后关系,该逻辑提交时间戳用于表示事务的逻辑提交时刻,该事务快照时间戳用于表示事务的全局开始时刻。

[0362] 在一种可能实施方式中,该确定存储单元用于:

[0363] 响应于数据库系统处于除了顺序并发结果可串行化隔离级别之外的可串行化隔离级别,对该至少一个待判断数据项中任一待判断数据项,若产生该待判断数据项的事务的全局提交时间戳小于该目标事务的事务快照时间戳,确定该待判断数据项为候选数据项;将具有相同主键标识的候选数据项中全局提交时间戳最大的候选数据项确定为目标数据项;

[0364] 其中,在该顺序并发结果可串行化隔离级别中任意两个并发事务的逻辑执行生命周期用于确定该两个并发事务的先后关系,该全局提交时间戳用于表示事务的全局提交时刻,该事务快照时间戳用于表示事务的全局开始时刻。

[0365] 在一种可能实施方式中,若该目标事务涉及针对数据项的写入操作,该执行模块802用于:

[0366] 根据该执行请求,生成待写入数据项,将该待写入数据项存储到该目标事务的写集中。

[0367] 在一种可能实施方式中,基于图8的装置组成,该装置还包括:

[0368] 回滚模块,用于响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,若该目标事务的写集中包括与该目标数据项具有相同主键标识的待写入数据项,回滚该目标事务。

[0369] 在一种可能实施方式中,该调整模块803用于:

[0370] 将该逻辑执行生命周期的时间戳下界调整至大于产生该目标数据项的事务的逻辑提交时间戳,该逻辑提交时间戳用于表示事务的逻辑提交时刻;

[0371] 将该逻辑执行生命周期的时间戳上界调整至小于或等于产生第一数据项的事务的逻辑提交时间戳,该第一数据项为与该目标数据项具有相同主键标识的下一数据项。

[0372] 在一种可能实施方式中,该调整模块803用于:

[0373] 响应于该目标数据项对应的待写事务不为空,将该逻辑执行生命周期的时间戳上界调整至小于或等于该待写事务的时间戳下界。

[0374] 在一种可能实施方式中,该调整模块803用于:

[0375] 将该逻辑执行生命周期的时间戳下界调整至大于或等于各个待写入数据项的最大读事务时间戳中的最大值,该最大读事务时间戳用于表示读取过该待写入数据项的各事务的逻辑提交时间戳中的最大值。

[0376] 在一种可能实施方式中,基于图8的装置组成,该调整模块803包括:

[0377] 第一调整单元,用于对该写集中任一待写入数据项的读事务列表中的任一读事务,响应于该读事务处于验证通过状态或提交完成状态,将该逻辑执行生命周期的时间戳下界调整至大于或等于该读事务的时间戳上界;

[0378] 第二调整单元,用于响应于该读事务处于正在运行状态,在不同的可串行化隔离级别下,对该逻辑执行生命周期进行不同的调整。

[0379] 在一种可能实施方式中,该第二调整单元用于:

[0380] 响应于数据库系统处于线性可串行化、线性并发结果可串行化或者顺序可串行化中任一隔离级别,在等待该读事务结束后继续处理该目标事务,将该逻辑执行生命周期的时间戳下界调整至大于或等于该读事务的时间戳上界;

[0381] 响应于数据库系统处于线性点可串行化隔离级别,回滚该目标事务;

[0382] 响应于数据库系统处于顺序并发结果可串行化隔离级别,将该逻辑执行生命周期的时间戳下界调整至大于或等于该读事务的时间戳下界,将该读事务的时间戳上界调整至小于或等于该逻辑执行生命周期的时间戳下界。

[0383] 在一种可能实施方式中,该逻辑执行生命周期的校验过程包括:

[0384] 响应于该逻辑执行生命周期的时间戳下界小于或等于该逻辑执行生命周期的时间戳上界,确定对该逻辑执行生命周期校验通过;

[0385] 响应于该逻辑执行生命周期的时间戳下界大于该逻辑执行生命周期的时间戳上界,确定对该逻辑执行生命周期校验不通过。

[0386] 上述所有可选技术方案,可以采用任意结合形成本公开的可选实施例,在此不再一一赘述。

[0387] 需要说明的是:上述实施例提供的事务处理装置在处理事务时,仅以上述各功能模块的划分进行举例说明,实际应用中,可以根据需要而将上述功能分配由不同的功能模块完成,即将计算机设备的内部结构划分成不同的功能模块,以完成以上描述的全部或者部分功能。另外,上述实施例提供的事务处理装置与事务处理方法实施例属于同一构思,其具体实现过程详见事务处理方法实施例,这里不再赘述。

[0388] 图9是本申请实施例提供的一种计算机设备的结构示意图。请参考图9,计算机设备900可以是协调节点设备,也可以是数据节点设备,该计算机设备900可因配置或性能不同而产生比较大的差异,可以包括一个或一个以上处理器(Central Processing Units, CPU) 901和一个或一个以上的存储器902,其中,该存储器902中存储有至少一条程序代码,该至少一条程序代码由该处理器901加载并执行以实现上述各个实施例提供的事务处理方法。当然,该计算机设备900还可以具有有线或无线网络接口、键盘以及输入输出接口等部件,以便进行输入输出,该计算机设备900还可以包括其他用于实现设备功能的部件,在此不做赘述。

[0389] 在示例性实施例中,还提供了一种计算机可读存储介质,例如包括至少一条程序代码的存储器,上述至少一条程序代码可由终端中的处理器执行以完成上述实施例中事务处理方法。例如,该计算机可读存储介质可以是ROM(Read-Only Memory,只读存储器)、RAM(Random-Access Memory,随机存取存储器)、CD-ROM(Compact Disc Read-Only Memory,只读光盘)、磁带、软盘和光数据存储设备等。

[0390] 本领域普通技术人员可以理解实现上述实施例的全部或部分步骤可以通过硬件来完成,也可以通过程序来指令相关的硬件完成,该程序可以存储于一种计算机可读存储介质中,上述提到的存储介质可以是只读存储器,磁盘或光盘等。

[0391] 以上所述仅为本申请的可选实施例,并不用以限制本申请,凡在本申请的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本申请的保护范围之内。

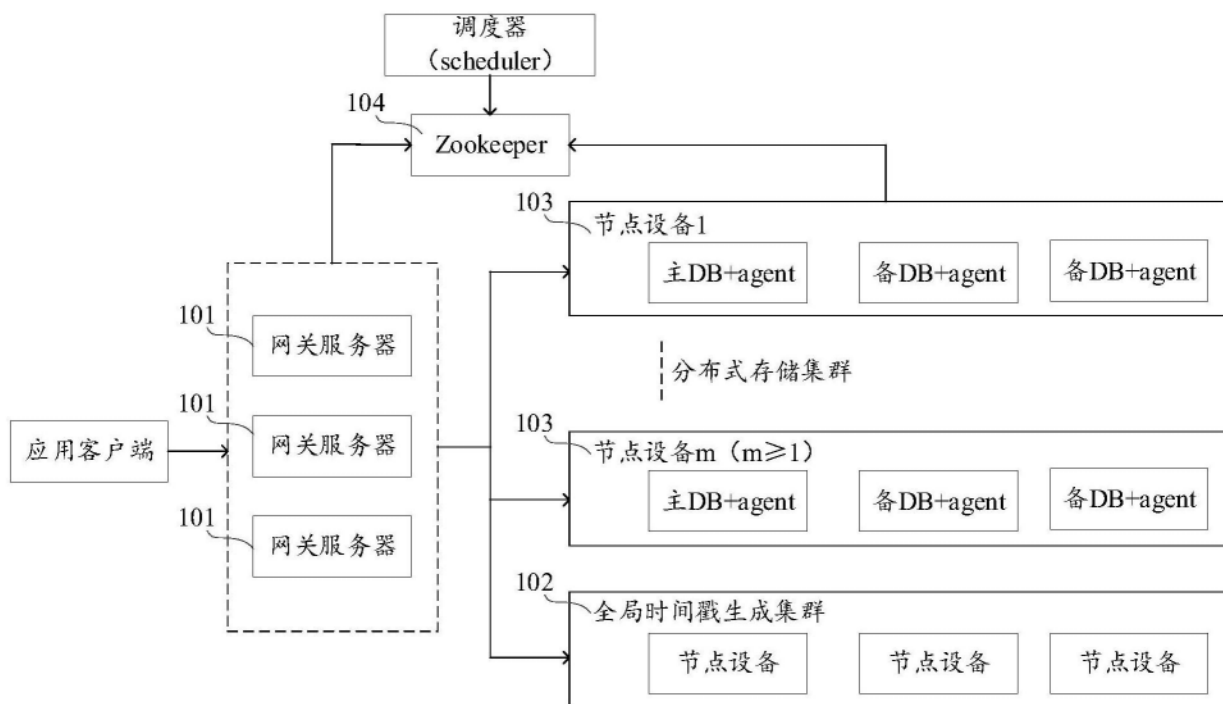


图1

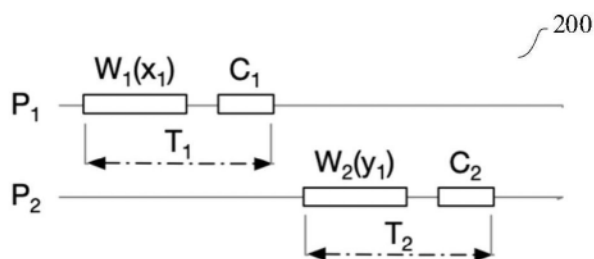


图2

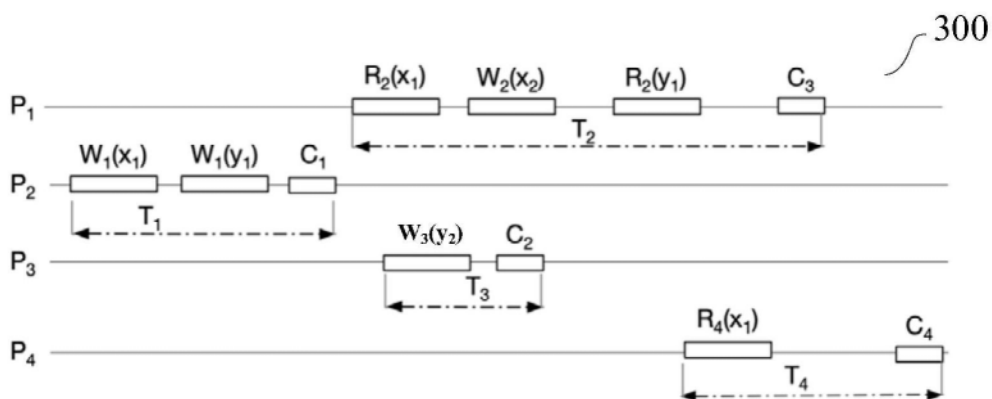


图3

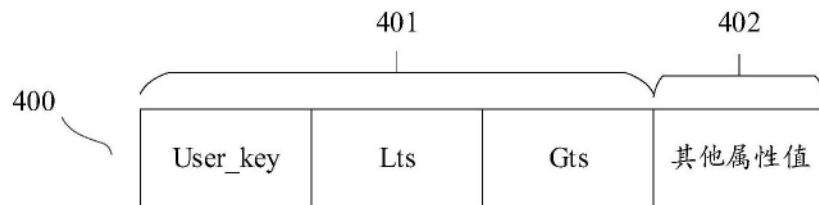


图4

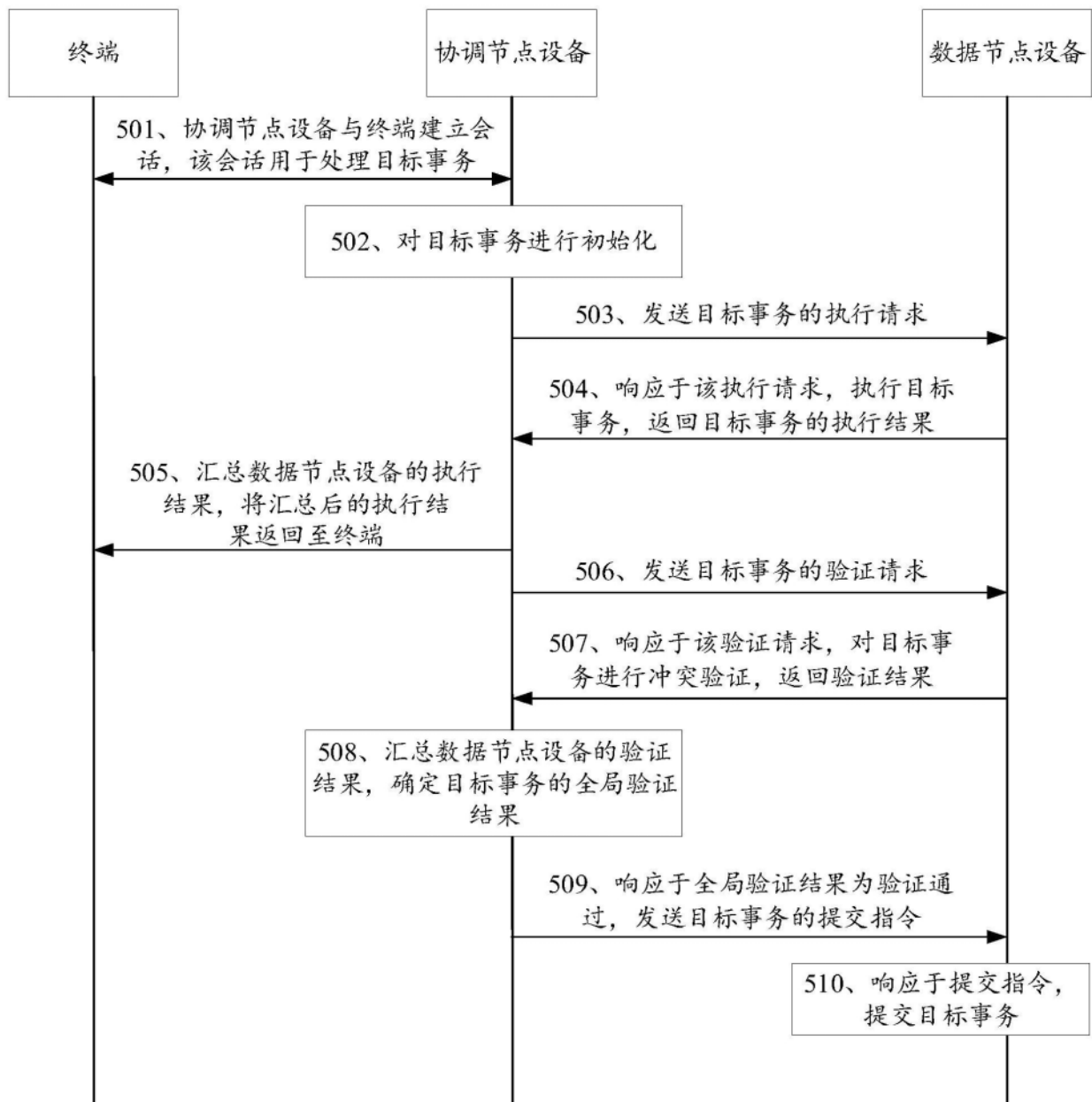


图5

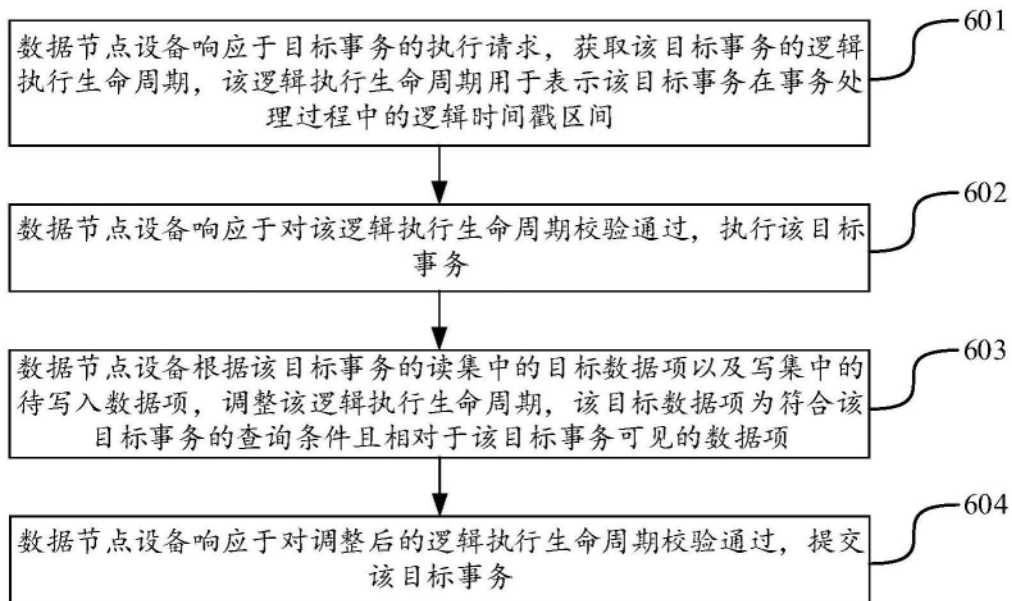


图6

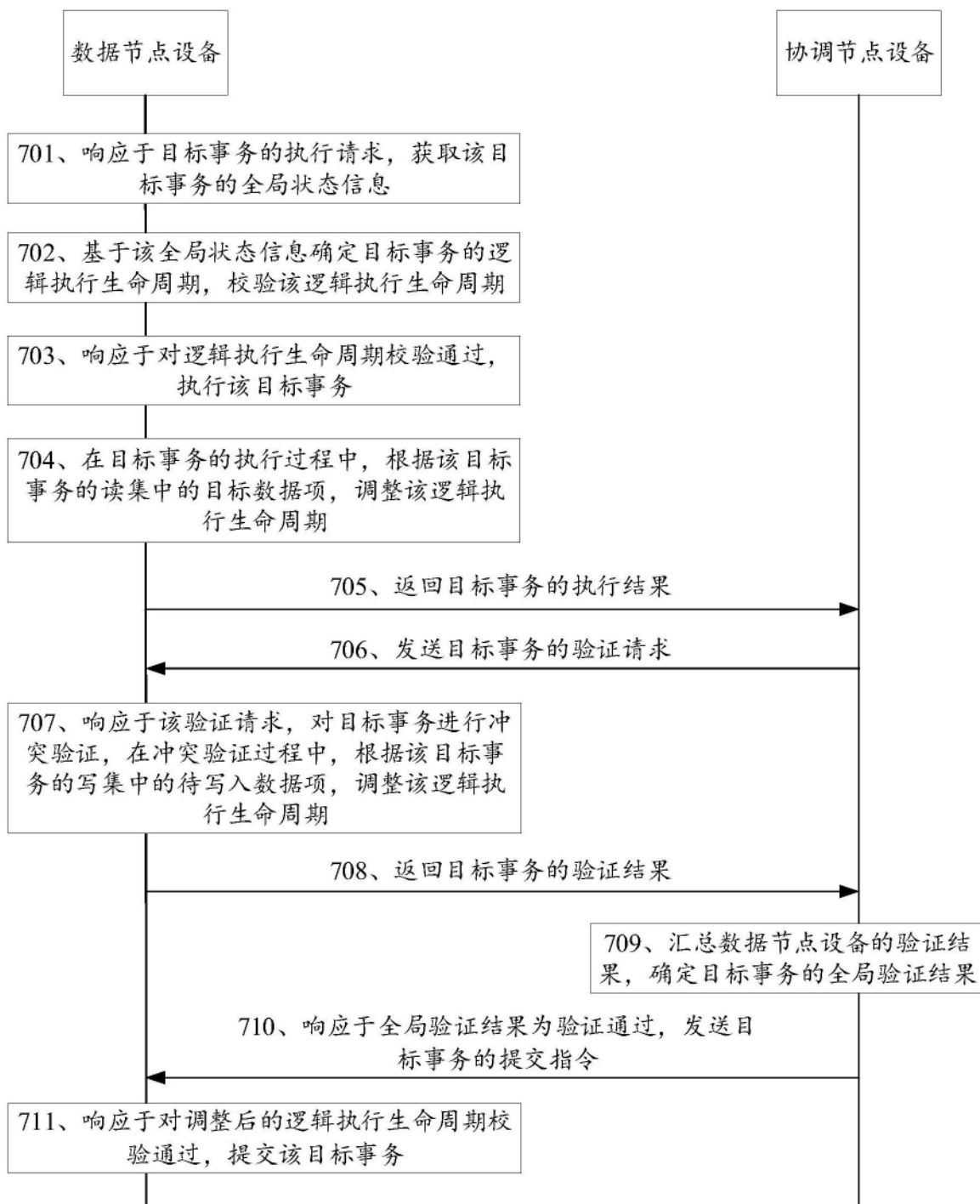


图7

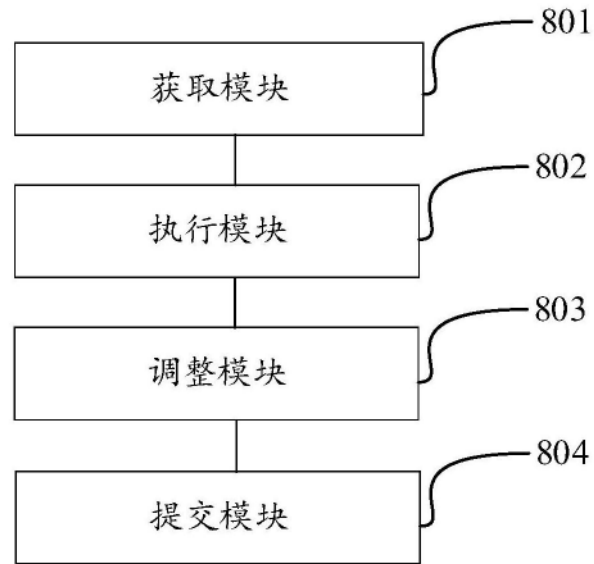


图8

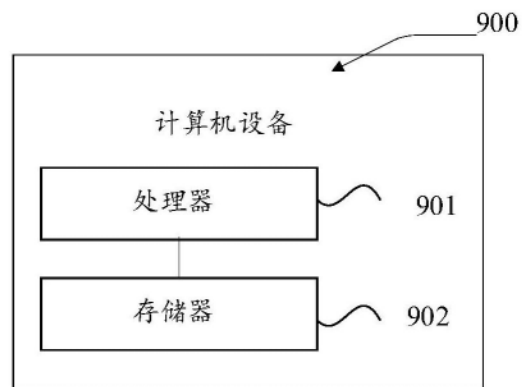


图9